Spatial Databases

Esteban ZIMÁNYI
Department of Computer & Decision Engineering (CoDE)
Université Libre de Bruxelles
ezimanyi@ulb.ac.be

ULB

Info-H-415 Advanced Databases
Academic Year 2012-2013

Spatial Databases: Topics

- ➥ **Introduction**
- Georeferences and Coordinate Systems
- Conceptual Modeling for Spatial Databases
- Logical Modeling for Spatial Databases
- SQL/MM
- Representative Systems
- Summary

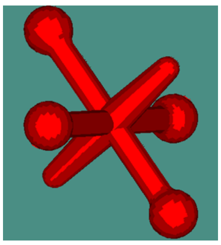
Foreword

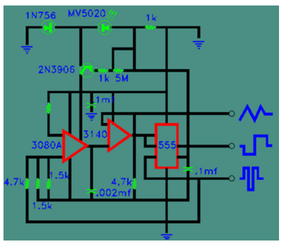
- In 1854 in Soho (London) doctor John Snow was able to identify the water point at the origin of the cholera epidemy by correlating the location of the water points and the location of the victims

Spatial Databases

- A database that needs to store and query spatial objects, e.g.
  - Point: a house, a monument
  - Line: a road segment, a road network
  - Polygon: a county, a voting area
- Types of spatial data


GIS Data


CAD Data



CAM Data

Spatial Database Management Systems

- A Database Management System that manages data existing in some space
- 2D or 2.5D
  - Integrated circuits: VLSI design
  - Geographic space (surface of the Earth): GIS, urban planning
- 2.5 D: Elevation
- 3D
  - Medicine: Brain models
  - Biological research: Molecule structures
  - Architecture: CAD
  - Ground models: Geology
- Supporting technology able to manage large collections of geometric objects
- Major commercial and open-source DBMSs provide spatial support

Why Spatial Databases?

- Queries to databases are posed in high level declarative manner (usually using SQL)
- SQL is popular in the commercial database world
- Standard SQL operates on relatively simple data types
- Additional spatial data types and operations can be defined in spatial database
- SQL was extended to support spatial data types and operations, e.g., OGC Simple Features for SQL
- A DBMS is a way of storing information in a manner that
  - Enforces consistency
  - Facilitates access
  - Allows users to relate data from multiple tables together

Application Areas

- Street network-based
  - Vehicle routing and scheduling (cars, planes, trains)
  - Location analysis, ...
- Natural resource-based
  - Management of areas: agricultural lands, forests, recreation resources, wildlife habitat analysis, migration routes planning...
  - Environmental impact analysis
  - Toxic facility siting
  - Groundwater modeling, ...
- Land parcel-based
  - Zoning, subdivision plan review
  - Environmental impact statements
  - Water quality management
  - Facility management: electricity, gaz, clean water, used water, network

Interaction with End Users

- Display data (e.g., maps) on the screen
- Access other data by clicking on it (hypermaps)
- Address queries
- Perform operations

Example of Queries

- On a subway map of Brussels, what is the shortest way from here to the Grand Place?
  ⇒ shortest path algorithm
- Overlay the land use map with the map of districts in Belgium
- Display today's weather forecast in the Brussels Region
- Given the map of a neighborhood, find the best spot for opening a drugstore (based on a given set of optimality criteria) ⇒ allocation problem

Geographic Information Systems

- A system designed to capture, store, manipulate, analyze, manage, and present geographically-referenced data as well as non-spatial data
- Connection between system elements is geography, e.g., location, proximity, spatial distribution
- Common purpose: decision making for managing use of land, resources, ocean data, transportation, geomarketing, urban planning, etc.
- Many commercial and open source systems, but limited temporal support

GIS as a Set of Subsystems

- Data processing
  - Data acquisition (from maps, images): input, store
- Data analysis
  - Retrieval, analysis (answers to queries, complex statistical analyses)
- Information use
  - Users: Researchers, planners, managers
  - Interaction needed between GIS group and users to plan analytical procedures and data structures
- Management system
  - Organizational role: separate unit in a resource management
  - Agency offering spatial database and analysis services
  - System manager, system operator, system analysts, digitizer operators

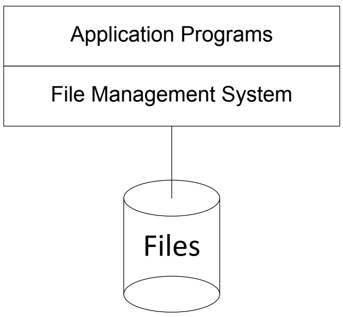
Contributing Disciplines and Technologies (1)

- Convergence of technological fields and traditional disciplines
- **Geography**: The study of the Earth
- **Cartography**: Display of spatial information
  - Computer cartography (digital, automated cartography): methods for digital representation, manipulation and visualization of cartographic features
- **Remote sensing**: Techniques for image acquisition and processing (space images)
- **Photogrammetry**: Aerial photos and techniques for accurate measurements
- **Geodesy**: The study of the size and shape of the earth
- **Statistics**: Statistical techniques for analysis + errors and uncertainty in GIS data
- **Operations Research**: Optimizing techniques for decision making
- **Mathematics**: (Computational) geometry and graph theory for analysis of spatial data
- **Civil Engineering**: Transportation, urban engineering

Contributing Disciplines and Technologies (2)

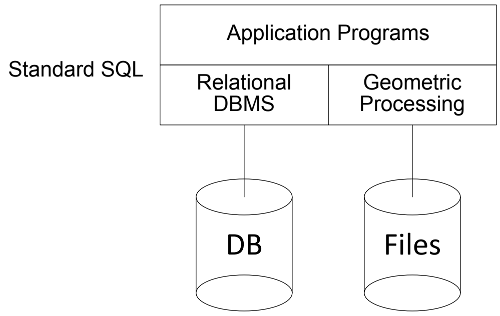
- **Computer Science**
  - Computer Aided Design (CAD): Software, techniques for data input, display, representation, visualization
  - Computer graphics: Hardware, software for handling graphic objects
  - Artificial Intelligence (AI): Emulate human intelligence and decision making (computer = "expert").
  - Database Management Systems (DBMS): Representing, handling large volumes of data (access, updates)

GIS Architectures: Ad Hoc Systems



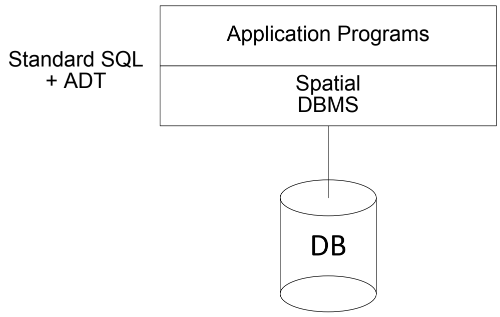
- Not modular
- Not reusable
- Not extensible
- Not friendly
- Very efficient

GIS Architectures: Loosely Coupled Approach



- Structured information and geometry stored at different places
  - Relational DBMS for alphanumerical (non spatial) data
  - Specific module for spatial data management
- Modularity (use of a DBMS) BUT:
  - Heterogeneous models! Difficult to model, integrate and use
  - Partial loss of basic DBMS functionalities e.g., concurrency, optimization, recovery, querying

GIS Architectures: Integrated Approach



- "Extended relational" system
- Modular
- Extensible (add a new operation, a new type)
- Reusable
- Friendly (easy change, easy use)

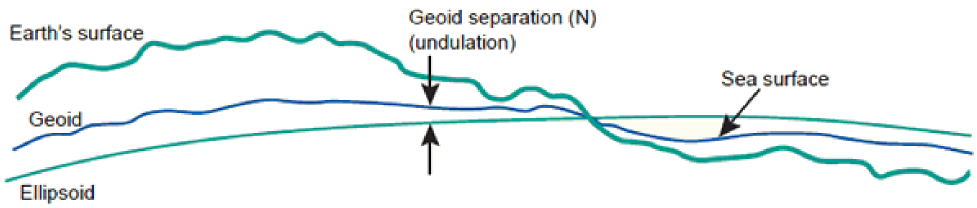
Main Database Issues

- Data modeling
- (Spatial) query language
- Query optimization
- Spatial access methods define index structure to accelerate the retrieval of objects
- Versioning
- Data representation (raster/vector)
- Graphical user interfaces (GUI)
- Computational geometry for GIS

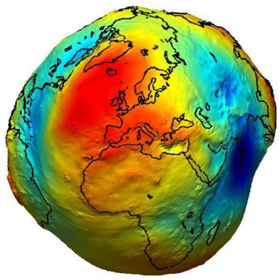
Spatial Databases: Topics

Projected Coordinates Systems

- Why focus on projections?
  - Going from 3 to 2 dimensions for Earth representation always involves a projection
  - Information on the projection is essential for applications analyzing spatial relationships
  - Choice of the projection can influence the results
- The Earth is a complex surface whose shape and dimensions can not be described with mathematical formulas
- Two main reference surfaces are used to approximate the shape of the Earth: the **ellipsoid** and the **Geoid**

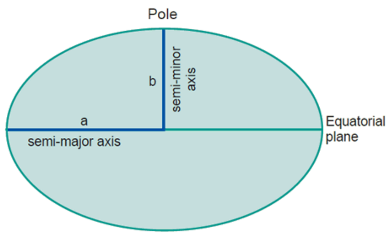


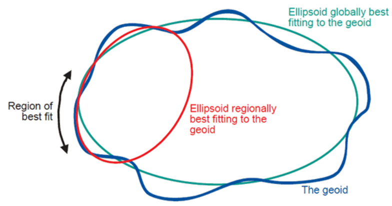
The Geoid



- Reference model for the physical surface of the Earth
- It is the equipotential surface of the Earth's gravity field which best fits the global mean sea level and extended through the continents
- Used in geodesy, e.g., for measuring heights represented on maps
- Not very practical to produce maps
- Since the mathematical description of the geoid is unknown, it is impossible to identify mathematical relationships for moving from the Earth to the map

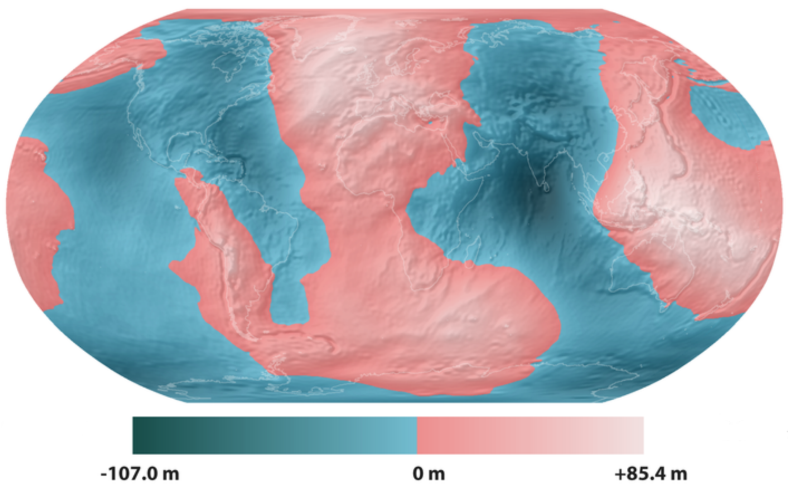
Ellipsoid





- A mathematically-defined surface that approximates the geoid
- Flattening $f = \frac{a-b}{a}$: how much the symmetry axis is compressed relative to the equatorial radius
- For the Earth, f is around 1/300: difference of the major and minor semi-axes of approx. 21 km
- Ellipsoid used to measure locations, the latitude (ϕ) and longitude (λ), of points of interest
- These locations on the ellipsoid are then projected onto a mapping plane
- Different regions of the world use different reference ellipsoids that minimize the differences between the geoid and the ellipsoid
- For Belgium (since 2008) ellipsoid GRS80 (=WGS84) with a = 6,378,137 m. and $f = 1/298.257222101$

Difference between the EGM96 Geoide and the WGS84 Ellipsoid

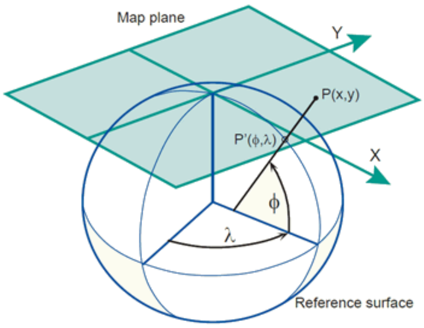


- The physical Earth has excursions of +8,000 m (Mount Everest) and -11,000 m (Mariana Trench)
- The geoid’s total variation goes from -107m to +85 m compared to a perfect mathematical ellipsoid

Latitude and Longitude

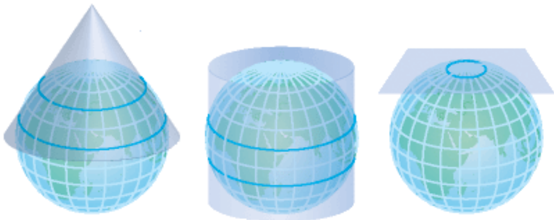
- Measures of the angles (in degrees) from the center of the Earth to a point on the Earth's surface
- Latitude measure angles in a North-South direction: the Equator is at an angle of 0
- Longitude measures angles in an East-West direction: the Prime Meridian is at angle of 0
- Prime Meridian: imaginary line running from the North to the South Pole through Greenwich, England
- For example, the location of Brussels, Belgium is 50.8411° N, 4.3564° E
- Longitude and latitude are not uniform units of measure
- Only along the Equator the distance represented by one degree of longitude approximates the distance represented by one degree of latitude

Map Projections



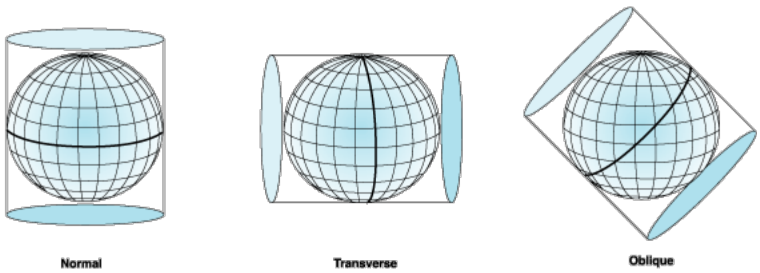
- To produce a map, the curved reference surface of the Earth, approximated by an ellipsoid or a sphere, is transformed into the flat plane of the map by means of a map projection
- A point on the reference surface of the Earth with geographic coordinates (ϕ, λ) is transformed into Cartesian (or map) coordinates (x, y) representing positions on the map plane
- Each projection causes deformations
- Map projections can be categorised in four ways: Class, Angle, Fit and Properties

Map Projections: Shape of Projection Surface



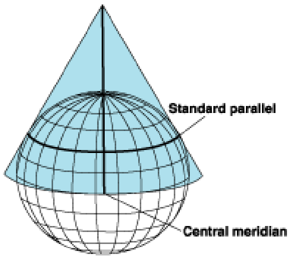
- Shape of the projection surface, commonly either a flat plane, a cylinder or a cone
- Cones and cylinders are not flat shapes, but they can be rolled flat without introducing additional distortion
  - **Cylindrical**: coordinates are projected onto a rolled cylinder
  - **Conical**: coordinates are projected onto a rolled cone
  - **Azimuthal**: coordinates are projected directly onto a flat planar surface
- Azimuthal projections work best for circular areas (e.g., the poles)
- Cylindrical projections work best for rectangular areas (e.g., world maps)
- Conical projections work best for triangle shaped areas (e.g., continents)

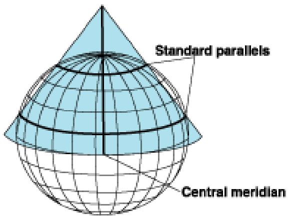
Map Projections: Angle

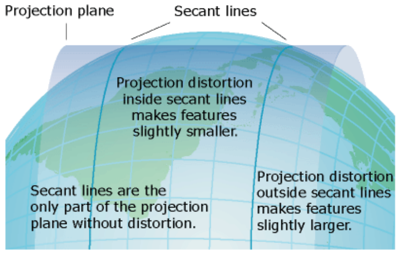


- This refers to the alignment of the projection surface, measured as the angle between the main axis of the earth and the main symmetry axis of the projection surface.
  - **Normal**: the two axes are parallel
  - **Transverse**: the two axes are perpendicular
  - **Oblique**: the two axes are at some other angle
- Ideally the plane of projection is aligned as closely as possible with the main axis of the area to be mapped. This helps to minimise distortion and scale error.

Map Projections: Fit





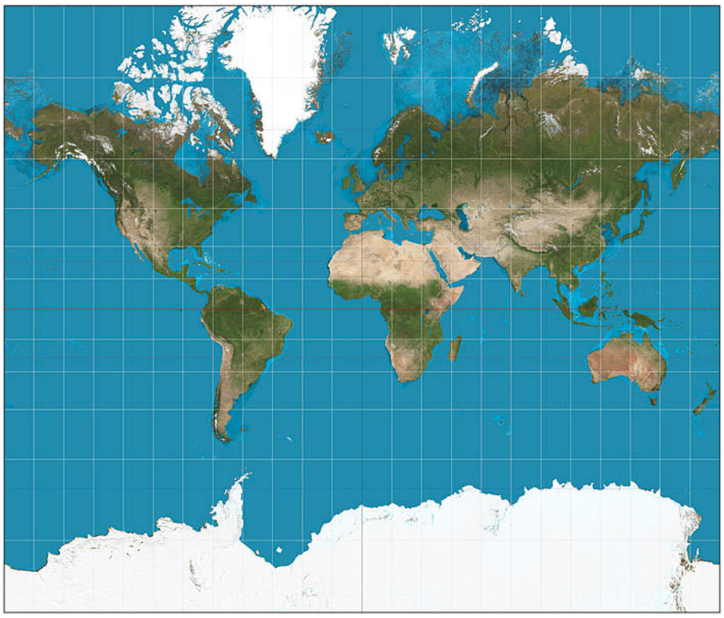


- A measure of how closely the projection surface fits the surface of the Earth
  - **Tangent**: the projection surface touches the surface of the Earth
  - **Secant**: the projection surface slices through the Earth
- Distortion occurs wherever the projection surface is not touching or intersecting the surface of the Earth
- Secant projections usually reduce scale error because the two surfaces intersect in more places and the overall fit tends to be closer
- A globe is the only way to represent the entire Earth without any significant scale error

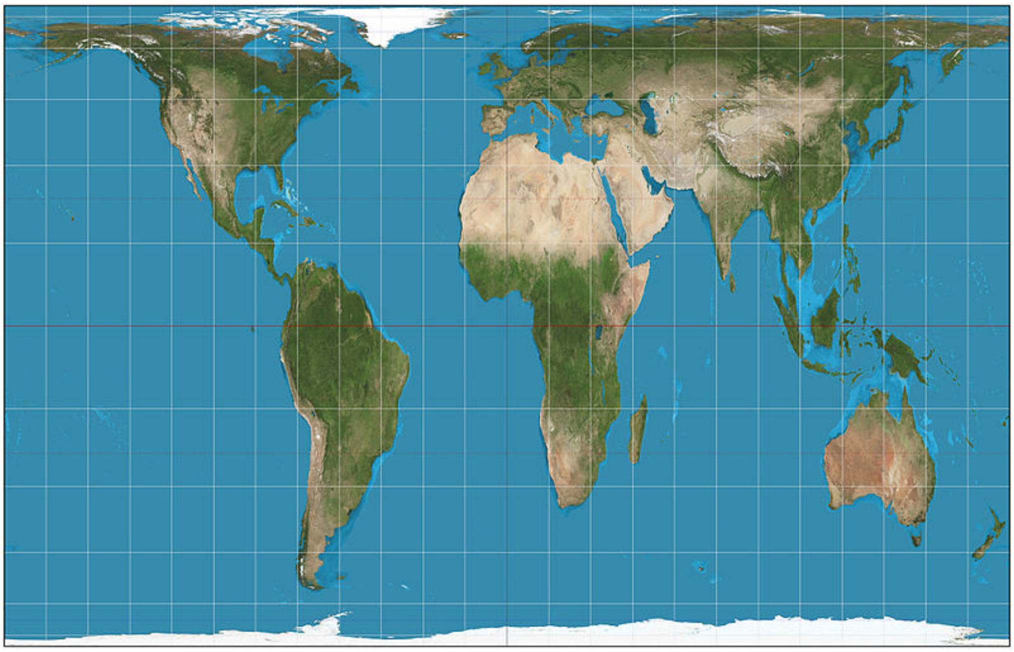
Map Projections: Geometric Deformations

- What is preserved
  - **Conformal**: preserve shapes and angles
  - **Equal Area** (or **equivalent**): preserve areas in correct relative size (shapes not preserved)
  - **Equidistant**: preserve distance (this is only possible at certain locations or in certain directions)
  - **True-direction** (or **Azimuthal**): preserves accurate directions (e.g., angles preserved, but length of lines is not)
- It is impossible to construct a map that is both equal-area and conformal
- Conformal map projections are recommended for navigational charts and topographic maps
- Equal area projections are generally best for thematic mapping
- Equidistant map projections should be used when measuring distances from a point: air routes, radio propagation strength, radiation dispersal

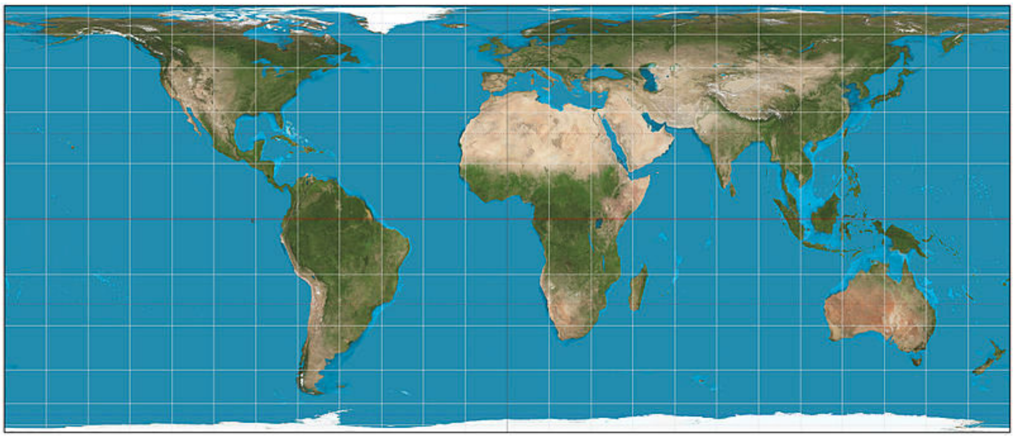
Conformal Cylindrical Projection: Mercator Projection



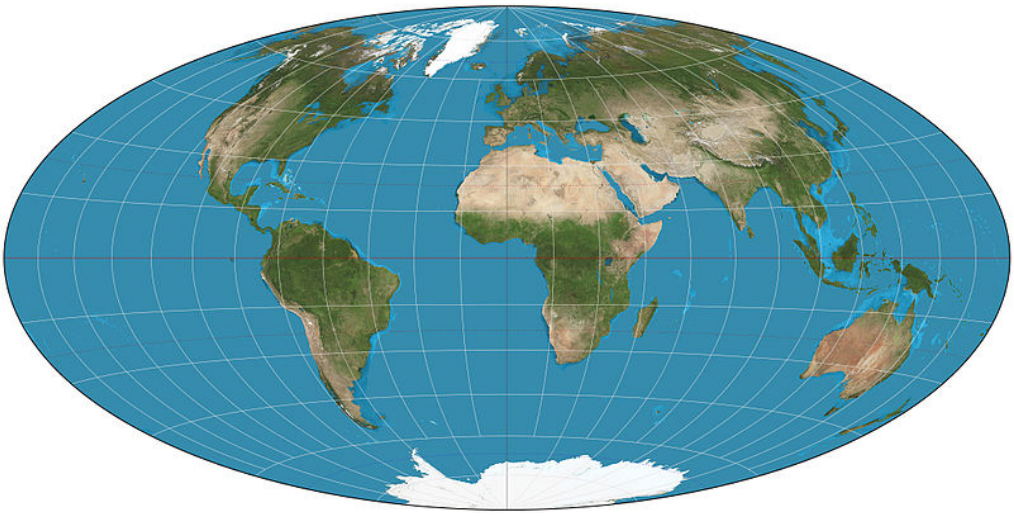
Cylindrical Equal-Area Projection: Peters Projection



Cylindrical Equal-Area Projection: Behrmann Projection

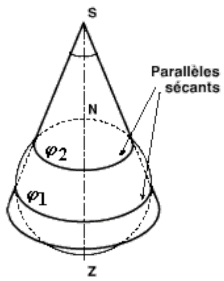


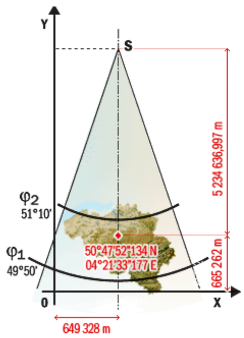
Elliptical Equal-Area projection: Hammer Projection



True-direction Azimuthal Projection

Belgium: Projection Lambert 2008





- Projection plane: Conical
- Direction of the projection axis: Normal
- Nature of the contact: Secant
- Nature of the deformation: Conformal

Parameters

Ellipsoid	Identity	GRS80
	semi-major axis a	6,378,137 m
	flattening f	1/298.257222101
Standard Parallels	φ_1	49° 50’ N
	φ_2	51° 10’ N
Origine	Origin Latitude	50° 47’ 52” 134 N
	Central meridian	4° 21’ 33” 177 E
Origin Coordinates	x_0	649,328 m
	y_0	665,262 m

Spatial Databases: Topics

- Introduction
- Georeferences and Coordinate Systems
- ➥ **Conceptual Modeling for Spatial Databases**
- Logical Modeling for Spatial Databases
- SQL/MM
- Representative Systems
- Summary

Data Modeling Requirements

- Multiple views of space
  - discrete (objects), continuous (fields)
  - 2D, 2.5D, 3D
- Multiple representations
  - different scales, different viewpoints
- Several spatial abstract data types
  - point, line, area, set of points, set of lines, set of areas, . . .
- Explicit spatial relationships
  - crossing, adjacency, . . .

Interaction Requirements

- Visual interactions
  - map displays
  - information visualization
  - graphical queries on maps
- Flexible, context-dependent interactions
- Multiple user profiles
  - highway: constructor, car driver, truck driver, hiker, ecologist
- Multiple instantiations
  - a building may be a school and a church
  - a road segment may be also a segment of a hiking trail

Practical Requirements

- Huge data sets
  - Collecting new data is expensive
  - Reusing highly heterogeneous existing data sets is a must ... but is very difficult!
  - Integration requires understanding, hence a conceptual model
- Integration of DB with different space/time granularity
- Coexistence with non-spatial, non-temporal data
- Reengineering of legacy applications
- Interoperability

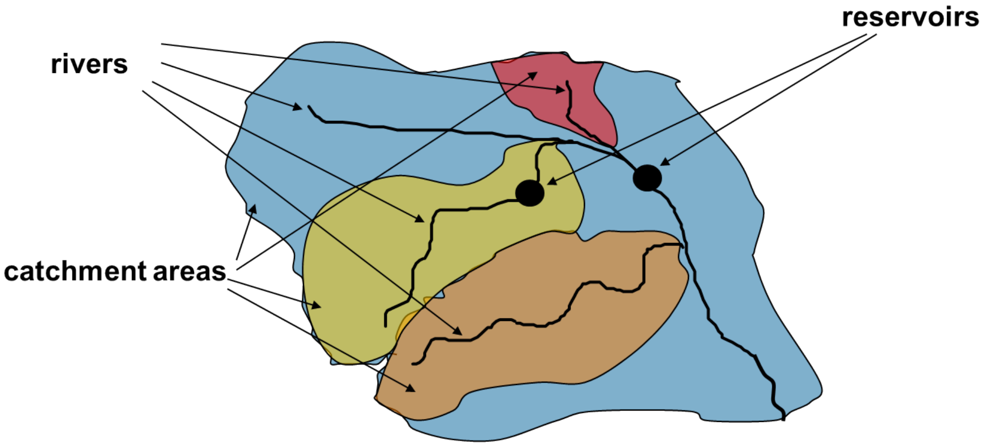
Why Conceptual Modeling?

- Focuses on the application
- Technology independent
  - portability, durability
- User oriented
- Formal, unambiguous specification
- Supports visual interfaces
  - data definition and manipulation
- Best vehicle for information exchange/integration

The Spatiotemporal Conceptual Manifesto

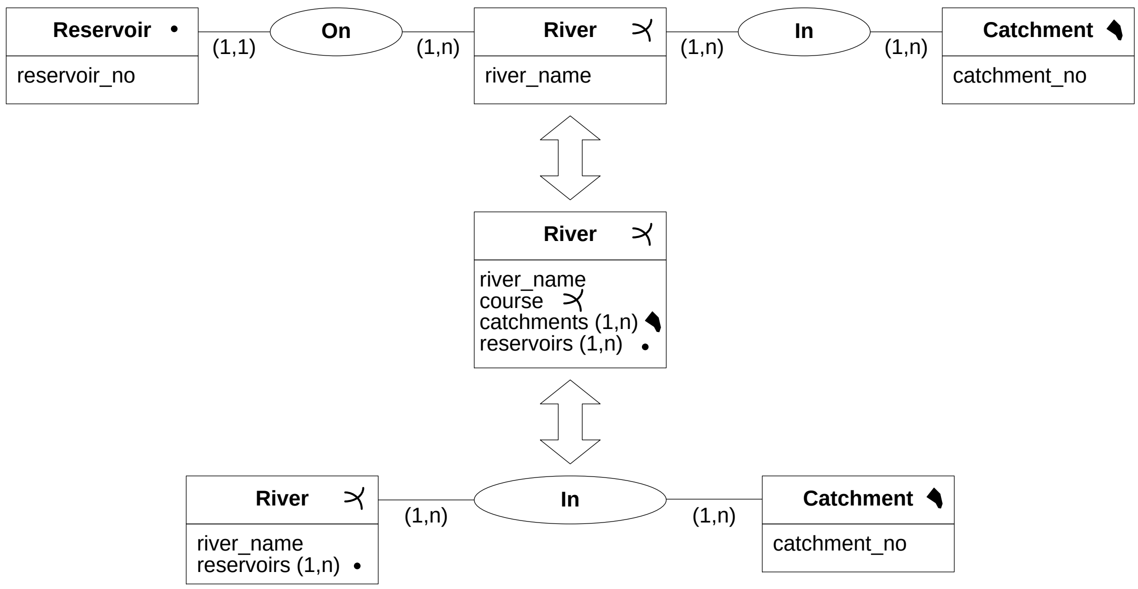
- Good expressive power
- Simple (understandable) data model
  - few clean concepts, with standard, well-known semantics
- No artificial constructs (e.g., space / time objects)
- Orthogonality of space, time and data structures
- Similarity of concepts for space and time
- Clean, visual notations and intuitive icons / symbols
- Formal definition
- Associated query language

Orthogonality: What is Spatial



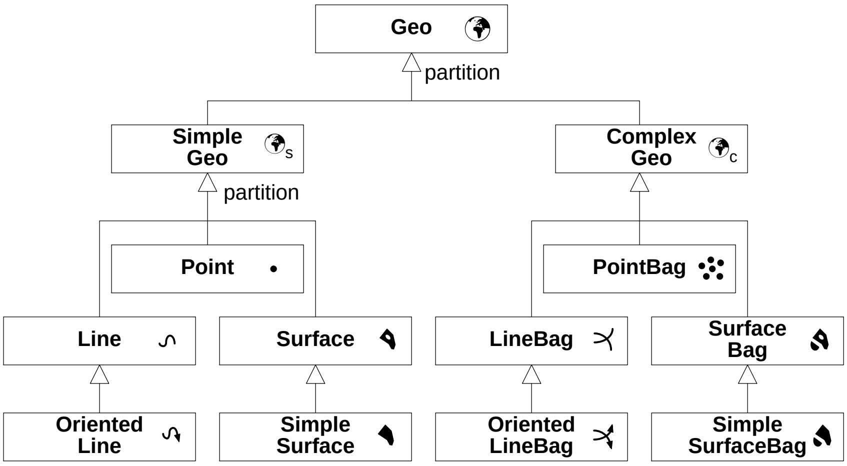
41

Same Space, Different Data Structures



42

MADS Spatial Data Types: Type Hierarchy



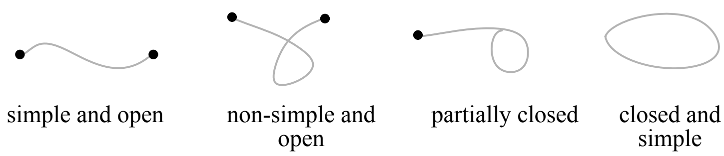
- Spatial data types are **topologically closed**: all geometries include their boundary
- `Geo`, `SimpleGeo`, and `ComplexGeo` are abstract classes

MADS Spatial Data Types (1)

Point

- Boundary of a point: empty set

Line

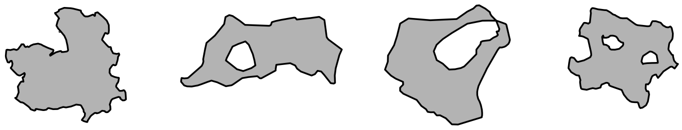


- Boundary of a line: the extreme points, if any

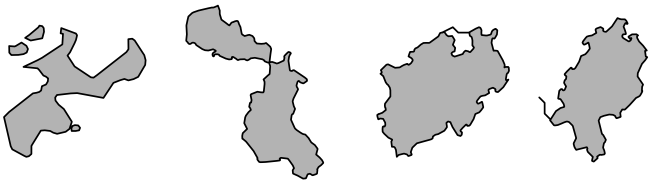
MADS Spatial Data Types (2)

Surface

- Defined by 1 exterior boundary and 0 or more interior boundaries defining its holes



- Examples of geometries that are not surfaces

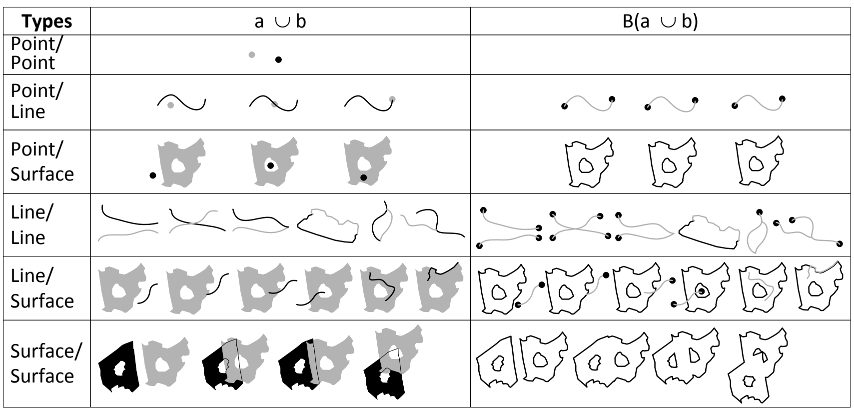


several surfaces   several surfaces   has a cut line   has a spike

Complex Geometries and Their Boundaries

- Boundary of a `ComplexGeo` value is defined (recursively) by the spatial union of
  (1) the boundaries of its components that do not intersect with other components
  (2) the intersecting boundaries that do not lie in the interior of their union

$$B(a \cup b) = B(a) - b \cup B(b) - a \cup ((B(a) \cap b) \cup (B(b) \cap a) - I(a \cup b))$$



Topological Predicates

- Specify how two geometries relate to each other
- Based on the definition of their **boundary**, **interior**, and **exterior**, denoted by $I(x)$, $B(x)$, and $E(x)$
- $Dim(x)$: maximum dimension (-1, 0, 1, or 2) of x, -1 corresponds to the dimension of the empty set
- Dimensionally extended 9-intersection matrix (DE-9IM) for defining topological predicates

	Interior	Boundary	Exterior
Interior	$Dim(I(a) \cap I(b))$	$Dim(I(a) \cap B(b))$	$Dim(I(a) \cap E(b))$
Boundary	$Dim(B(a) \cap I(b))$	$Dim(B(a) \cap B(b))$	$Dim(B(a) \cap E(b))$
Exterior	$Dim(E(a) \cap I(b))$	$Dim(E(a) \cap B(b))$	$Dim(E(a) \cap E(b))$

- Dense notation use a string of 9 characters to represent the cells of the matrix
- Possible characters: T (non-empty intersection), F (empty intersection), 0, 1, 2, * (irrelevant)
- Example: a and b are disjoint if their intersection is empty

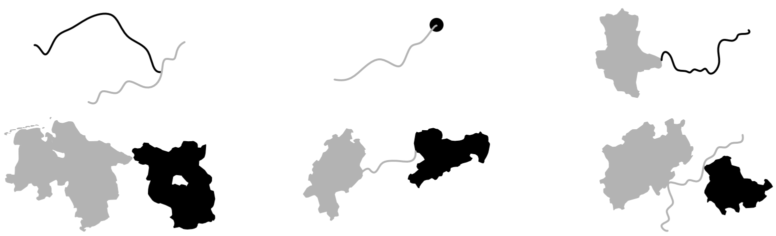
$$I(a) \cap I(b) = \emptyset \wedge I(a) \cap B(b) = \emptyset \wedge B(a) \cap I(b) = \emptyset \wedge B(a) \cap B(b) = \emptyset$$

corresponds to ‘FF*FF****’

Topological Predicates (1)

Meets

- a meets $b \Leftrightarrow I(a) \cap I(b) = \emptyset \wedge a \cap b \neq \emptyset \wedge Dim(a \cap b) = 0$

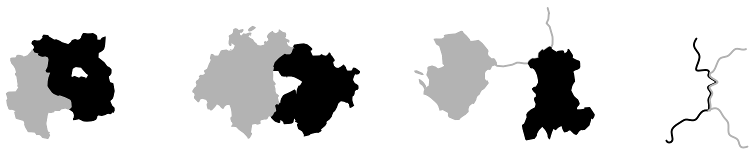


- Examples of geometries that do not satisfies the meets predicate

Topological Predicates (2)

Adjacent

- a adjacent $b \Leftrightarrow I(a) \cap I(b) = \emptyset \wedge a \cap b \neq \emptyset \wedge Dim(a \cap b) = 1$



- The last example does not satisfies the predicate, their intersection is at the interior of both geometries

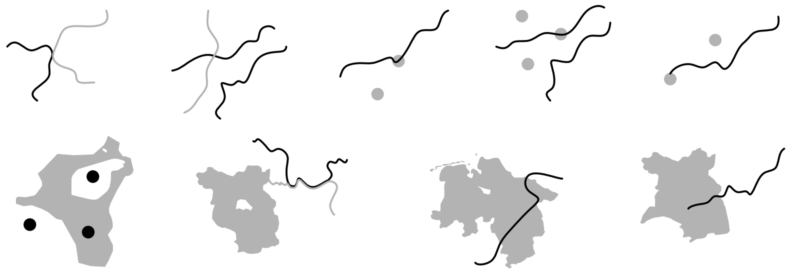
Touches

- a touches $b \Leftrightarrow I(a) \cap I(b) = \emptyset \wedge a \cap b \neq \emptyset \Leftrightarrow a$ meets $b \vee a$ adjacent b

Topological Predicates (3)

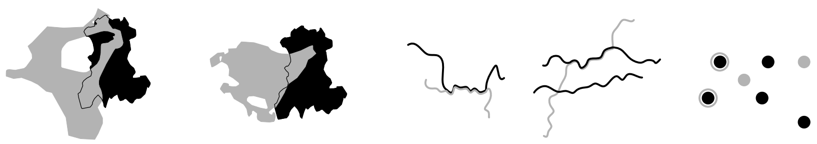
Crosses

- a crosses $b \Leftrightarrow Dim(I(a) \cap I(b)) < max(Dim(I(a)), Dim(I(b))) \wedge a \cap b \neq a \wedge a \cap b \neq b \wedge a \cap b \neq \emptyset$



Overlaps

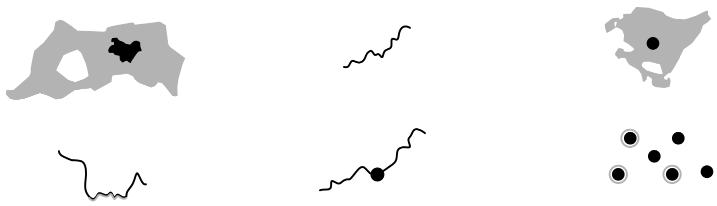
- a overlaps $b \Leftrightarrow Dim(I(a)) = Dim(I(b)) = Dim(I(a) \cap I(b)) \wedge a \cap b \neq a \wedge a \cap b \neq b$



Topological Predicates (4)

Contains/Within

- a contains $b \Leftrightarrow I(a) \cap I(b) \neq \varnothing \wedge a \cap b = b \Leftrightarrow b$ within a



Disjoint/Intersects

- a disjoint $b \Leftrightarrow a \cap b = \varnothing \Leftrightarrow \neg a$ intersects b

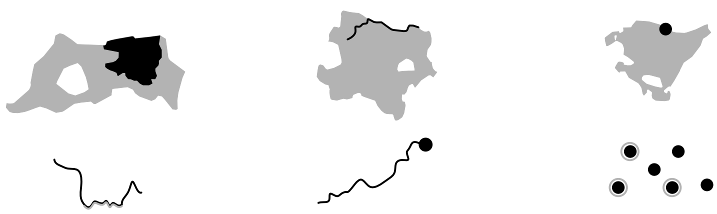
Equals

- a equals $b \Leftrightarrow a \cap b = a \wedge a \cap b = b \Leftrightarrow (a - b) \cup (b - a) = \varnothing$

Topological Predicates (5)

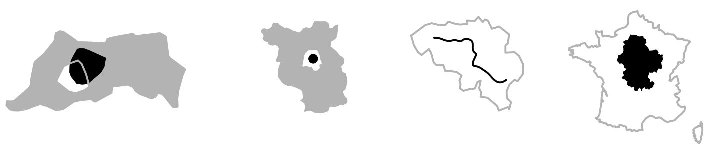
Covers

- a covers $b \Leftrightarrow a \cap b = b \Leftrightarrow b - a = \varnothing$



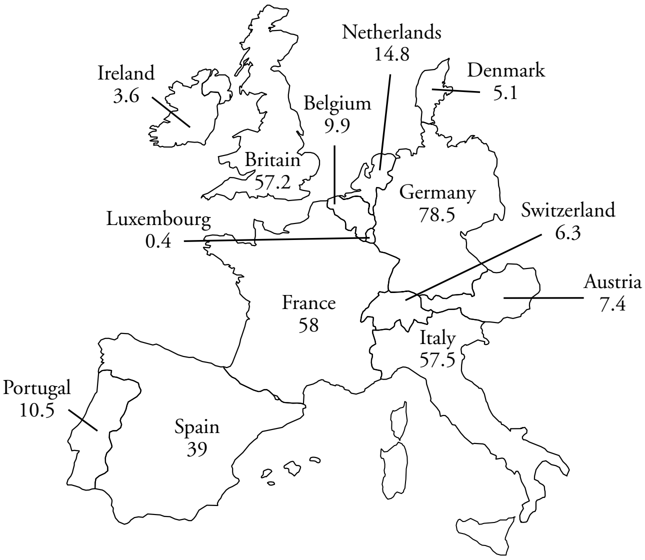
Encloses/Surrounded

- Definition is quite involved, depends of whether a is a (set of) line(s) or a (set of) surface(s)



Spatial Objects

Country
name population

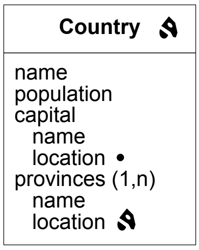


Spatial Attributes

Client	Country	Road
name address location •	name population capital • rivers (1,n)	name responsible stations (1,n) •

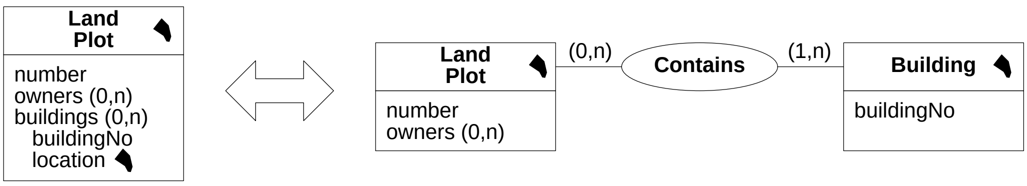
- Both non-spatial and spatial object types can have spatial attributes
- Domain of spatial attribute: a spatial type (point, line, surface, . . .)
- Spatial attributes can be multi-valued
- A spatial attribute of a spatial object type **may** induce a topological constraint
  - The capital of a country is located within the geometry of its country
- This is not necessarily the case
  - A given country will keep **either** the full geometry of the rivers flowing through it **or** these geometries will be projected to the section flowing through the country
  - This depends on application semantics
- The conceptual schema must explicitly state these topological constraints

Spatial Complex Attributes



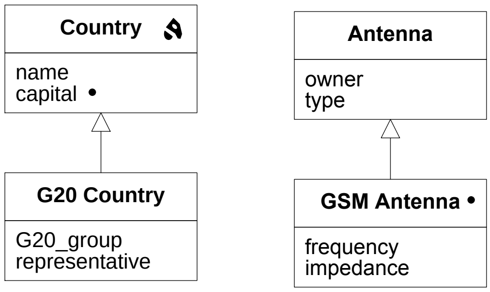
- Spatial attributes can be a component of a complex and/or multivalued attribute
- It is usual to keep both thematic (alphanumeric) and location data for attributes, e.g., capital
- This will allow to print both the name and the location of capitals/rivers/roads/... in a map
- However, in real maps the toponyms (names of objects appearing in a map) have also a location
  - There are precise cartographic rules for placing them, this is a semi-automatic process

Spatial Objects vs. Spatial Attributes



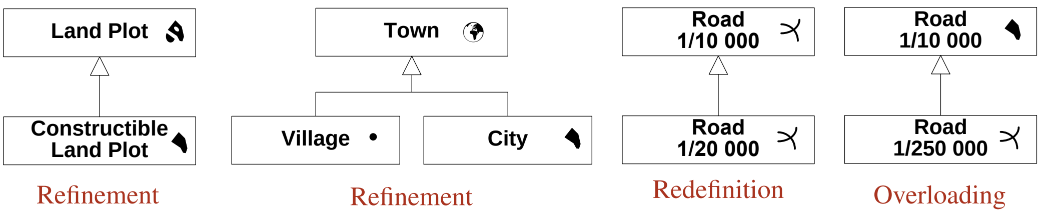
- Representing a concept as a spatial object or as a spatial attribute depends on the application
- Determined by the relative importance of the concept
- Has implications in the way of accessing the instances of the concept (e.g. the buildings)
  - As spatial objects: the application can access a building one by one
  - As spatial attributes: the access to a building must be made through the land plot containing it

Generalization: Inheriting Spatiality



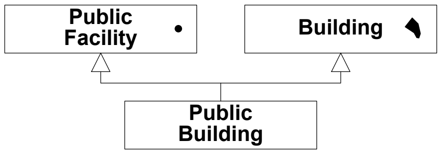
- Spatiality is inherited through generalization
  - As any other feature: attributes, methods, relationships, integrity constraints, etc.
- Based on the well-known **susbtitutability principle** in OOP
- For simple inheritance it is not necessary to restate the geometry in the subtype (but see later)
- As usual, spatiality can be added to a subtype
  - Only instances of the subtype will have associated spatiality

Generalization: Refining/Redefining Spatiality



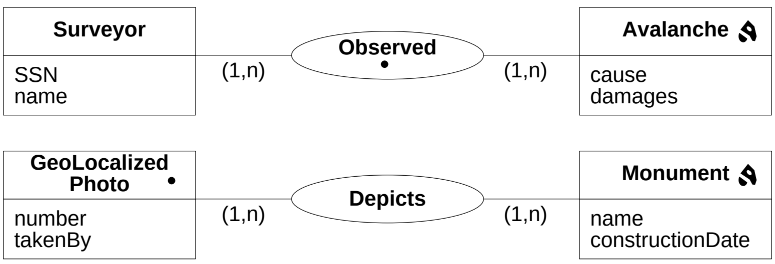
- Arise when a spatiality is restated in a subtype
- **Refinement**: restricts the inherited property, value remains the same in the supertype and the subtype
- In redefinition and overloading an instance of the subtype has both a locally defined spatiality and the inherited one
- **Redefinition**: Keeps substitutability wrt typing, allows dynamic binding
- **Overloading**: Relaxes substitutability, inhibiting dynamic binding

Generalization: Multiple Inheritance



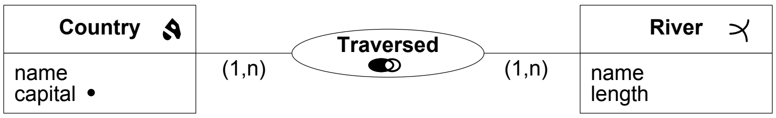
- Spatiality is inherited from several supertypes
  - Similar situation may arise with any kind of attribute
- There is ambiguity when referring to the spatiality of the subtype
- Several policies have been proposed for solving this issue in the OO community
- Most general policy
  - All inherited properties are available in the subtype, user must disambiguate in queries
  - `PublicFacility.geometry` vs `Building.geometry` for an instance of `PublicBuilding`

Spatial/Non-Spatial Relationships



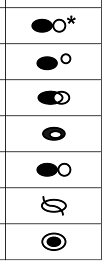
- Spatiality can also be defined for relationship types
- Spatiality of relationship types is orthogonal to the fact that linked object types are spatial
- If a spatial relationship type relates spatial type(s), spatial constraints **may** restrict the geometries

Topological Relationships

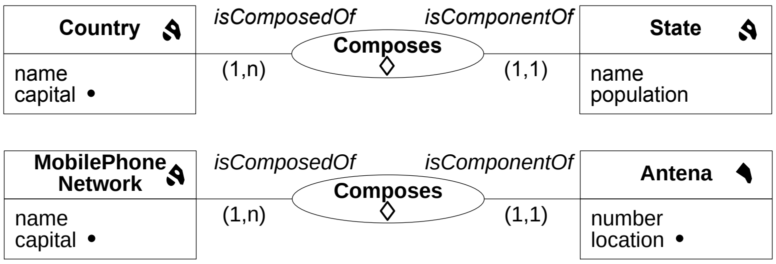


- Specified on a relationship type that links at least two spatial types
- Constrain the spatiality of the instances of the related types
- Many topological constraints can be defined using the 9IM
  - 5 between complex points
  - 8 between simple regions
  - 18 between simple regions with holes
  - 33 between complex regions
  - 43 between a complex line and a complex region
  - ...
- Conceptual model depicts only the most general ones

Topological Relationship	Icon
TopoGeneric	
TopoDisjoint	
TopoOverlap	
TopoWithin	
TopoTouch	
TopoCross	
TopoEqual	

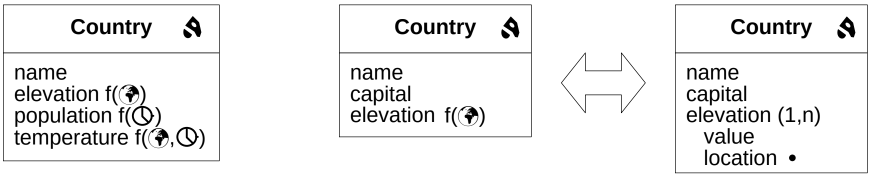


Spatial Aggregation



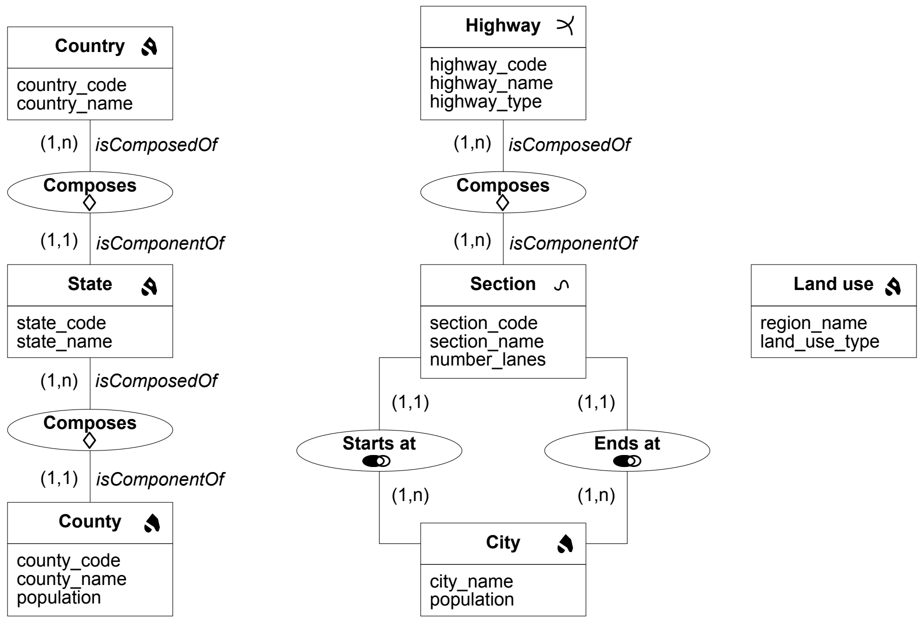
- Traditional aggregation relationship can link spatial types
- Usually, aggregation has exclusive semantics (stated by cardinalities in the component role)
- Usually, the spatiality of the aggregation is **partitioned** into the spatiality of the components
- It is not the case for the second example, where the spatiality of `Antena` corresponds to its coverage
  - The same location can be covered by several antennas
  - Spatiality of the aggregation is the **spatial union** of the spatiality of the antennas

Space- and Time-Varying Attributes



- Also referred to as continuous fields
- Allow to represent phenomena that change in space and/or in time
  - `elevation`: to each point in space is associated a real number
  - `population`: to each point in time is associated an integer number
  - `temperature`: to each point in space is associated a real number, this value evolves over time
- At the **conceptual level**, can be represented as continuous function
  - Operators for manipulating fields can be defined at this level
- At the **logical level** can be implemented in several ways
  - **Raster**: Discretize the space into regular cells, assign a value to each cell
  - **TIN**: Keep values at particular locations, interpolation used for calculating value at any point

Conceptual Schema: Example



Spatial Databases: Topics

Representation Models

- Representation of infinite point sets of the Euclidean space in a computer
- Two alternative representations
- **Object-based models** (Vector)
  - Describes the spatial extent of relevant objects with a set of points
  - Uses points, lines, and surfaces for describing spatiality
  - Choice of geometric types is arbitrary, varies across systems
- **Field-based models** (Raster)
  - Each point in space is associated with one/several attribute values, defined as continuous functions
  - Examples: altitude, temperature, precipitation, polution, etc.

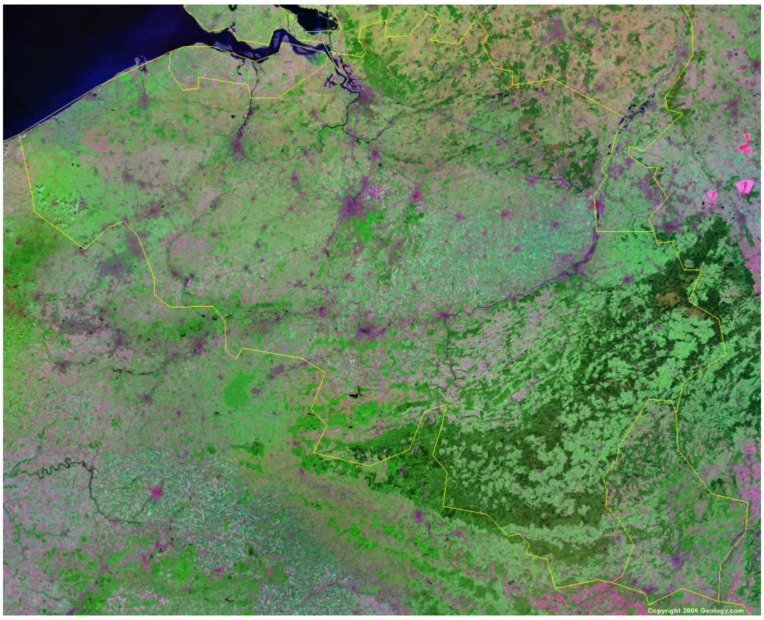
Belgium Map: Vector

67

Belgium Map: Raster

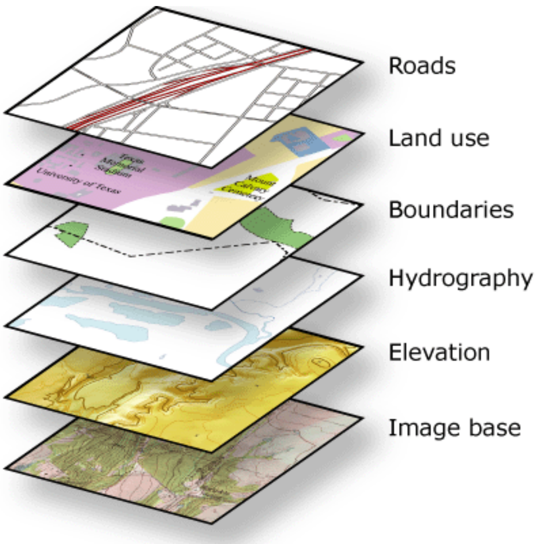
68

Belgium Map: Satellite



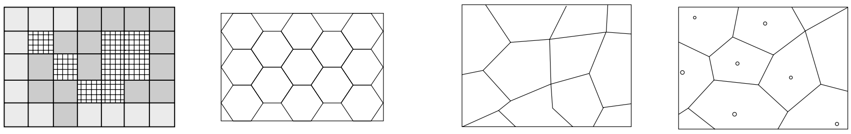
69

Organizing Spatial Data: Layers



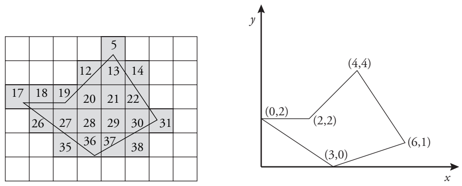
Raster Model: Tesselation

- Decomposition of the plane into polygonal units
- May be **regular** or **irregular**, depending on whether the polygonal units are of equal size



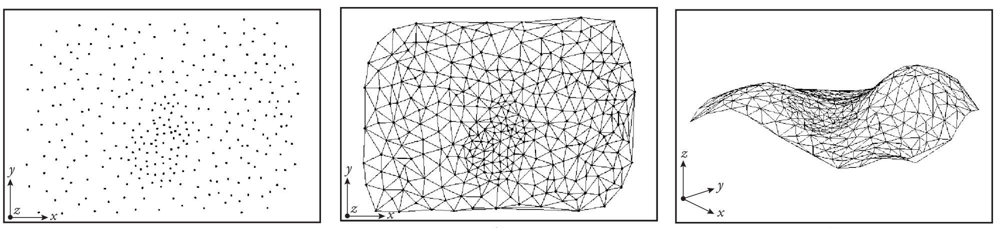
Regular Irregular

- Regular tesselation used for remote sensing data (e.g., satellite images)
- Irregular tesselation used for zoning in social, demographic or economic data
- A spatial object is represented by the smallest subset of pixels that contains it



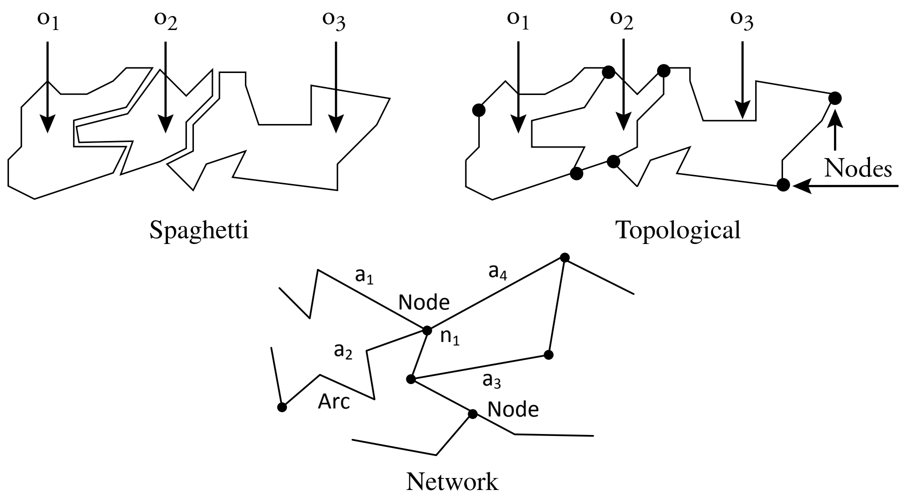
Digital Elevation Models (DEMs)

- Provide a digital (finite) representation of an abstract modeling of space
- DEMs are useful to represent natural phenomemon that is a continuous function of the 2D space
  - temperature, presure, moisture, or slope
- Based on a finite collection of sample values, values at other points are obtained by **interpolation**
- **Triangulated Irregular Networks** (TINs) are based on a triangular partition of the 2D space



- No assumption is made on the distribution and location of the vertices of the triangles
- Elevation value is recorded at each vertex
- Value at any other point P inferred by linear interpolation of the 3 vertices of the triangle containing P

Representing the Geometry of a Collection of Objects



- Three comonly used representations: Spaghetti, Network, Topological
- Mainly differ in the expression of topological relationships among the component objects

Spaghetti Model

- Geometry of any object described independently of other objects
- No topology is stored in the model, all topological relationships must be computed on demand
- Implies representation redundancy
  - E.g., boundary of adjacent regions represented twice
- Enables heterogeneous representations mixing points, polylines and regions without restrictions
  - E.g., polylines may intersect without the intersection points stored explicitly in the database
- Advantages
  - Simplicity
  - Provides the end user with easy input of new objects into the collection
- Drawbacks
  - Lack of explicity information about topological relationships among spatial objects
  - No sharing of information ⇒ redundancy, problem with large data sets, inconsistency

Network Model

- Destined for network (graph)-based applications
  - transportation services, utility management (electricity, telephone, ...)
- Topological relationships among points and polylines are stored
- **Nodes**: Distinguished point that connects a list of arcs
- **Arcs**: Polyline that starts at a node and ends at a node
- Nodes allow efficent line connectivity tests and network computations (e.g., shortest paths)
- Two types of points: **regular points** and **nodes**
- Depending on the implementation, the network is planar or nonplanar
- **Planar network**: each edge intersection is recorded as a node, even if does not correspond to a real-world entity
- **Nonplanar network**: edges may cross withoug producing an intersection
  - Examples include ground transportation with tunnels and passes

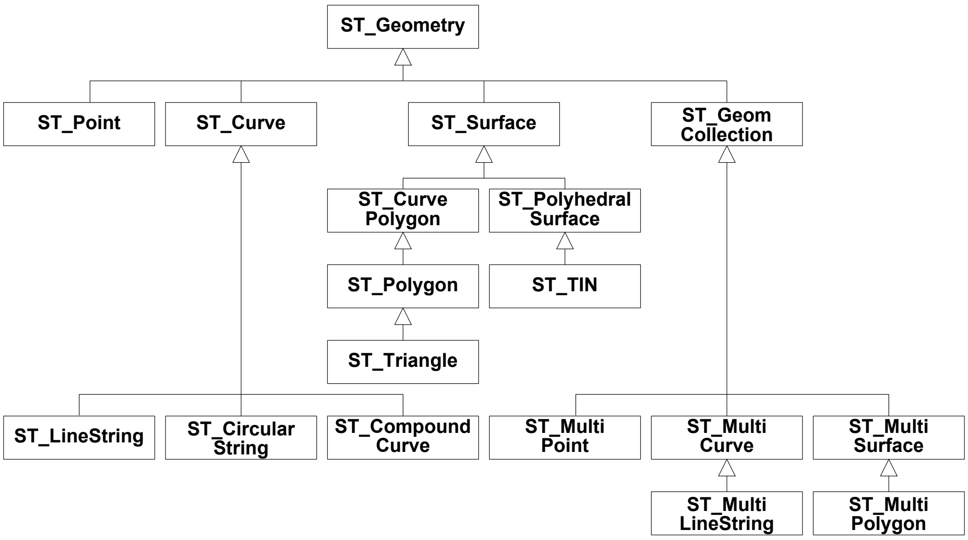
Topological Model

- Similar to the network model, except that the network is plannar
- Induces a planar subdivision into adjacent polygons, some of which may not correspond to actual geographic objects
- **Node**: represented by a point and the (possibly empty) list of arcs starting/ending at it
  - **Isolated point**: identifies location of point features such as towers, point of interest, ...
- **Arc**: features its ending points, list of vertices and two polygons having the arc as common boundary
- **Polygon**: represented by a list of arcs, each arc being shared with a neighbor polygon
- **Region**: represented by one or more adjacent polygons
- No redundacy: each point/line is stored only once
- **Advantages**: Efficient computation of topological queries, update consistency
- **Drawbacks**: Some database objects have no semantics in real-world, complexity of the structure may slow down some operations

Spatial Databases: Topics

- Introduction
- Georeferences and Coordinate Systems
- Conceptual Modeling for Spatial Databases
- Logical Modeling for Spatial Databases
- ➥ **SQL/MM**
- Representative Systems
- Summary

SQL/MM Spatial: Geometry Type Hierarchy



- `ST_Geometry`, `ST_Curve`, and `ST_Surface` are not instantiable types

ST_Geometry

- Represent 0D, 1D, and 2D geometries that exist in 2D ($\mathbb{R}^2$), 3D ($\mathbb{R}^3$) or 4D coordinate space ($\mathbb{R}^4$)
- Geometries in $\mathbb{R}^2$ have points with (x, y) coordinate values
- Geometries in $\mathbb{R}^3$ have points with either (x, y, z) or (x, y, m) coordinate values
- Geometries in $\mathbb{R}^4$ have points with (x, y, z, m) coordinate values
- The z coordinate of a point typically represent altitude
- The m coordinate of a point representing arbitrary measurement: key to supporting linear networking applications such as street routing, transportation, pipeline, ...
- Geometry values are topologically closed (they include their boundary)
- All locations in a geometry are in the same spatial reference system (SRS)
- Geometric calculations are done in the SRS of the first geometry in the parameter list of a routine
- If a routine returns a geometry or measurement (e.g., length or area), the value is in the SRS of the first geometry in the parameter list

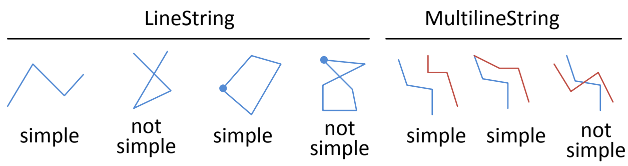
Methods on ST_Geometry[a]: Metadata (1)

- `ST_Dimension`: returns the dimension of a geometry
- `ST_CoordDim`: returns the coordinate dimension of a geometry
- `ST_GeometryType`: returns the type of the geometry as a `CHARACTER VARYING` value
- `ST_SRID`: observes and mutates the spatial reference system identifier of a geometry
- `ST_Transform`: returns the geometry in the specified spatial reference system
- `ST_IsEmpty`: tests if a geometry corresponds to the empty set

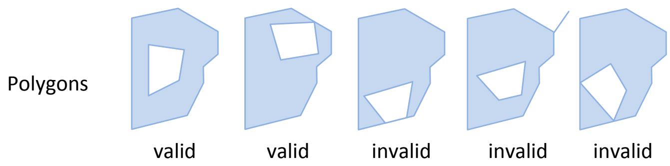
[a] 3D versions of some of these methods exists

Methods on ST_Geometry: Metadata (2)

- ST_IsSimple: tests if a geometry has no anomalous geometric points



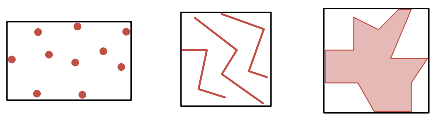
- ST_IsValid: tests if a geometry is well formed



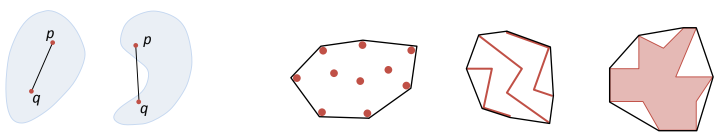
- ST_Is3D: tests whether a geometry has *z* coordinates
- ST_IsMeasured: tests whether a geometry has *m* coordinate values

Methods on ST_Geometry: Spatial Analysis (1)

- ST_Boundary: returns the boundary of a geometry
- ST_Envelope: returns the bounding rectangle of a geometry



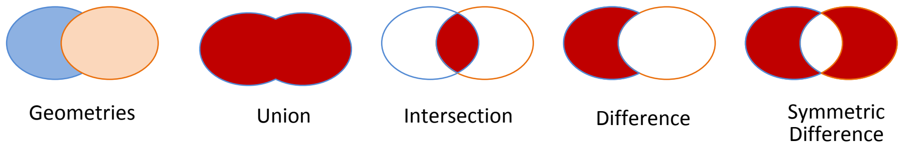
- ST_ConvexHull: returns the convex hull of a geometry



- ST_Buffer: returns the geometry that represents all points whose distance from any point of a geometry is less than or equal to a specified distance

Methods on ST_Geometry: Spatial Analysis (2)

- ST_Union: returns the geometry that represents the point set union of two geometries
- ST_Intersection: returns the geometry that represents the point set intersection of two geometries
- ST_Difference: returns the geometry that represents the point set difference of two geometries
- ST_SymDifference: returns the geometry that represents the point set symmetric difference of two geometries



- ST_Distance: returns the distance between two geometries

Methods and Functions on ST_Geometry: Input/Output

- ST_WKTToSQL: returns the geometry for the specified well-known text representation
- ST_AsText: returns the well-known text representation for the specified geometry
- ST_WKBToSQL: returns the geometry for the specified well-known binary representation
- ST_AsBinary: returns the well-known binary representation for the specified geometry
- ST_GMLToSQL: returns the geometry for the specified GML representation
- ST_AsGML: returns the GML representation for the specified geometry
- ST_GeomFromText: returns a geometry, which is transformed from a CHARACTER LARGE OBJECT value that represents its well-known text representation
- ST_GeomFromWKB: returns a geometry, which is transformed from a BINARY LARGE OBJECT value that represents its well-known binary representation
- ST_GeomFromGML: returns a geometry, which is transformed from a CHARACTER LARGE OBJECT value that represents its GML representation

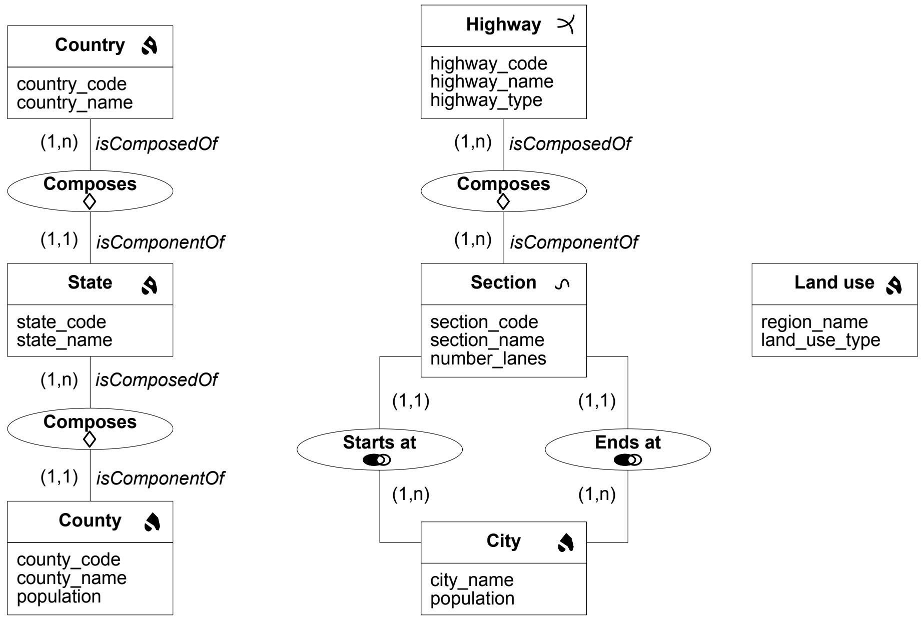
Boundary, Interior, Exterior

- **Boundary** of a geometry: set of geometries of the next lower dimension
  - `ST_Point` or `ST_MultiPoint` value: empty set
  - `ST_Curve`: start and end `ST_Point` values if nonclosed, empty set if closed
  - `ST_MultiCurve`: `ST_Point` values that are in the boundaries of an odd number of its element `ST_Curve` values
  - `ST_Polygon` value: its set of linear rings
  - `ST_MultiPolygon` value: set of linear rings of its `ST_Polygon` values
  - Arbitrary collection of geometries whose interiors are disjoint: geometries drawn from the boundaries of the element geometries by application of the mod 2 union rule
  - The domain of geometries considered consists of those values that are topologically closed
- **Interior** of a geometry: points that are left when the boundary points are removed
- **Exterior** of a geometry: points not in the interior or boundary

Spatial Relationships

- `ST_Equals`: tests if a geometry is spatially equal to another geometry
- `ST_Disjoint`: tests if a geometry is spatially disjoint from another geometry
- `ST_Intersects`: tests if a geometry spatially intersects another geometry
- `ST_Touches`: tests if a geometry spatially touches another geometry
- `ST_Crosses`: tests if a geometry spatially crosses another geometry
- `ST_Within`: tests if a geometry is spatially within another geometry
- `ST_Contains`: tests if a geometry spatially contains another geometry
- `ST_Overlaps`: tests if a geometry spatially overlaps another geometry
- `ST_Relate`: tests if a geometry is spatially related to another geometry by testing for intersections between their interior, boundary and exterior as specified by the intersection matrix
  - `a.ST_Disjoint(b)` $\Leftrightarrow (I(a) \cap I(b) = \emptyset) \wedge (I(a) \cap B(b) = \emptyset) \wedge$ $(B(a) \cap I(b) = \emptyset) \wedge (B(a) \cap B(b) = \emptyset) \Leftrightarrow$ `a.ST_Relate(b,'FF*FF****')`

Conceptual Schema: Example



Reference Schemas (1)

```
Create Table Country
(country_code integer,
country_name varchar (30),
geometry ST_MultiPolygon,
Primary Key (country_code))

Create Table State
(state_code integer,
state_name varchar (30),
country_code integer,
geometry ST_MultiPolygon,
Primary Key (state_code),
Foreign Key (country_code) References Country)

Create Table County
(county_code integer
county_name varchar (30),
state_code integer,
population integer,
geometry ST_MultiPolygon,
Primary Key (county_code),
Foreign Key (state_code) References State)
```

Reference Schemas (2)

```
/* Table Highway is NOT spatial */
Create Table Highway
(highway_code integer,
highway_name varchar (4),
highway_type varchar (2),
Primary Key (highway_code))

Create Table HighwaySection
(section_code integer,
section_number integer,
highway_code integer,
Primary Key (section_code,highway_code),
Foreign Key (section_code) References Section,
Foreign Key (highway_code) References Highway)
```


Reference Schemas (3)

```
Create Table Section
(section_code integer,
section_name varchar (4),
number_lanes integer,
city_start varchar (30),
city_end varchar (30),
geometry ST_Line,
Primary Key (section_code),
Foreign Key (city_start) References City,
Foreign Key (city_end) References City)

Create Table City
(city_name varchar (30),
population integer,
geometry ST_MultiPolygon,
Primary Key (city_name))

Create Table LandUse
(region_name varchar (30),
land_use_type varchar (30),
geometry ST_Polygon,
Primary Key (region_name))
```

Reference Queries: Alphanumerical Criteria (1)

- Number of inhabitants in the county of San Francisco

```
select population
from County
where county_name = 'San Francisco'
```

- List of the counties of the State of California

```
select county_name
from County, State
from State.state_code = County.state_code
and state_name = 'California'
```

- Number of inhabitants in the US

```
select sum (c2.population)
from Country c1, State s, County c2
where c1.country_name = 'USA'
and c1.country_code = s.country_code
and s.state_code = c2.state_code
```


Reference Queries: Alphanumerical Criteria (2)

- Number of lanes in the first section of Interstate 99

```
select s.number_lanes
from Highway h1, HighwaySection h2, Section s
where h1.highway_code = h2.highway_code
and h2.section_code = s.section_code
and h1.highway_name = 'I99'
and h2.section_number = 1
```

- Name of all sections that constitute Interstate 99

```
select s.section_name
from Highway h1, HighwaySection h2, Section s
where h1.highway_name = 'I99'
and h1.highway_code = h2.highway_code
and h2.section_code = s.section_code
```

Reference Queries: Spatial Criteria (1)

- Counties adjacent to the county of San Francisco in the same state

```
select c1.county_name
from County c1, County c2
where c2.county_name = 'San Francisco'
and c1.state_code = c2.state_code
and ST_Touches(c1.geometry, c2.geometry)
```

- Display of the State of California (supposing that the `State` table is no spatial)

```
select ST_Union(c.geometry)
from County c, State s
where s.state_code = c.state_code
and s.state_name = 'California'
```


Reference Queries: Spatial Criteria (2)

- Counties larger than the largest county in California

```
select c1.county_name
from County c1
where ST_Area(c1.geometry) >
    (select max (ST_Area(c.geometry))
     from County c, State s
     where s.state_code = c.state_code
     and s.state_name = 'California')
```

- Length of Interstate 99

```
select sum (ST_Length(s.geometry))
from Highway h1, HighwaySection h2, Section s
where h1.highway_name = 'I99'
and h1.highway_code = h2.highway_code
and h2.section_code = s.section_code
```

Reference Queries: Spatial Criteria (3)

- All highways going through the State of California

```
select distinct h1.highway_name
from State s1, Highway h1, HighwaySection h2, Section s2
where s1.state_name = 'California'
and h1.highway_code = h2.highway_code
and h2.section_code = s2.section_code
and ST_Overlaps(s2.geometry, s1.geometry)
```

- Display of all residential areas in the county of San Jose

```
select ST_Intersection(l.geometry, c.geometry)
from County c, LandUse l
where c.county_name = 'San Jose'
and l.land_use_type = 'residential area'
and ST_Overlaps(l.geometry, c.geometry)
```

- Overlay the map of administrative units and land use

```
select county_name, land_use_type, ST_Intersection(c.geometry, l.geometry)
from County c, LandUse l
where ST_Overlaps(c.geometry, l.geometry)
```


Reference Queries: Interactive Queries (1)

- Description of the county pointed to on the screen

```
select county_name, population
from County
where ST_Contains(geometry, @point)
```

- Counties that intersect a given rectangle on the screen

```
select county_name
from County
where ST_Overlaps(geometry, @rectangle)
```

- Part of counties that are within a given rectangle on the screen (clipping)

```
select ST_Intersection(geometry, @rectangle)
from County
where ST_Overlaps(geometry, @rectangle)
```

Reference Queries: Interactive Queries (2)

- Description of the highway section pointed to on the screen

```
select section_name, number_lanes
from Section
where ST_Contains(geometry, @point)
```

- Description of the highway(s) of which a section is pointed to on the screen

```
select h1.highway_name, h1.highway_type
from Highway h1, HighwaySection h2, Section s
where h1.highway_code = h2.highway_code
and h2.section_code = s.section_code
and ST_Contains(s.geometry, @point)
```


SQL/MM: Conclusion

- SQL/MM provides a standard way to declare and manipulate geometries
- The last version includes 3D and 4D types
- Several spatial data type organized in a hierarchy with associated methods
- These methods can be combined in SQL queries and programs with standard ones
- We only convered a small part of the standard
  - For additional information refer to the document
- However, systems deviate, sometimes considerably from the standard

Spatial Databases: Topics

- Introduction
- Georeferences and Coordinate Systems
- Conceptual Modeling for Spatial Databases
- Logical Modeling for Spatial Databases
- SQL/MM
- Representative Systems
  - **Oracle**
- Summary

Oracle Locator

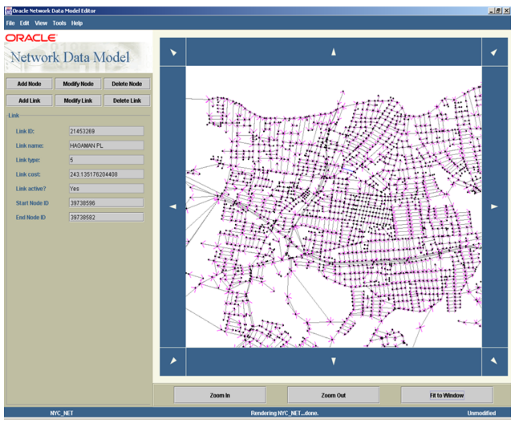
- Included in all editions of the database
- All functions required for standard GIS tools
- All geometric objects
  - Points, lines, polygons
  - 2D, 3D, 4D
- Indexing: quadtrees and rtrees
- Geometric queries
- Proximity search
- Distance calculation
- Multiple projections
- Conversion of projections

Oracle Spatial

- Advanced functions: Option of Oracle Database Enterprise Edition
- = Locator + ...
  - Geometric transformations
  - Spatial aggregations
  - Dynamic segmentation
  - Measures
  - Network modeling
  - Topology
  - Raster
  - Geocoder
  - Spatial Data Mining
  - 3D Types (LIDAR, TINS)
  - Web Services (WFS, CSW, OpenLS)

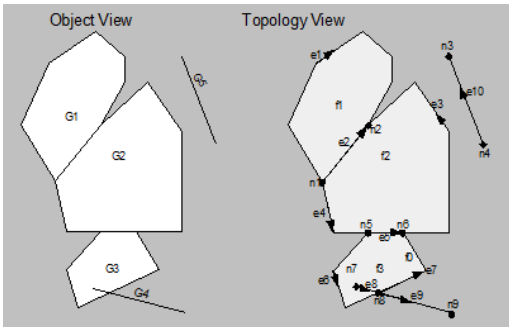
Oracle Network Model

- A data model for representing networks in the database
- Maintains connectivity
- Attributes at the link and node levels
- Used for network management
  - Transportation, logistics, utilities, location-based services, . . .
- Navigation engine for route calculation



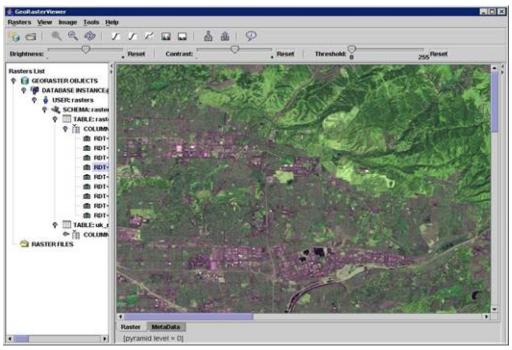
Oracle Topological Model

- Persistent storage of the topology
  - Nodes, arcs, faces
  - Topological relations
  - Allows advanced consistency checks
- Data model
  - Defining objects (features) through topological primitives
  - New type SDO_TOPO_GEOMETRY
  - Use traditional operators (SDO_RELATE ...)
- Co-existence with traditional spatial data
  - Possibility of combining SDO_GEOMETRY and SDO_TOPO_GEOMETRY

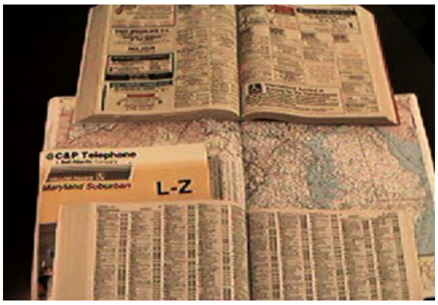


Oracle Geo Raster

- New data type SDO_GEORASTER
  - Satellite images, remote sensing data
  - Multi-band, multi-layer
  - Metadata in XML
  - Geo-referencing information
- Functionality
  - Open, general purpose raster data model
  - Storage and indexing of raster data
  - Querying and analyzing raster data
  - Delivering GeoRaster to external consumers: Publish as JPEG, GIFF images



Oracle Geocoding

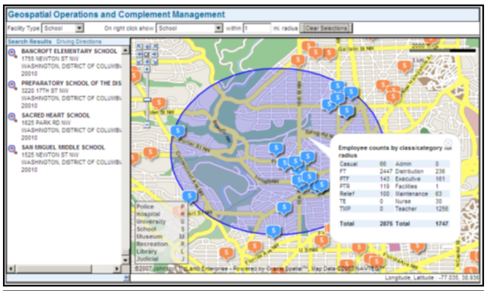


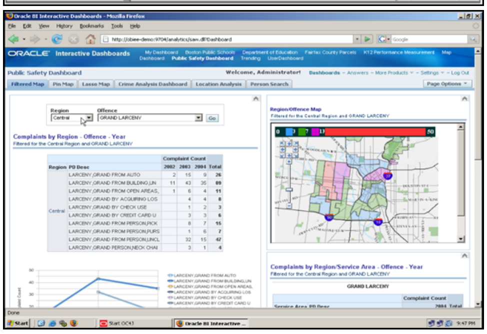
- Generates latitude/longitude (points) from address
- International addressing standardization
- Formatted and unformatted addresses
- Tolerance parameters support fuzzy matching
- Record-level and batch processes
- Data available from Navteq, TeleAtlas

105

Oracle MapViewer

- Supplied with all versions of Oracle Application Server
- XML interfaces, Java and Javascript (Ajax)
- Tool for map definition
- Maps described in the database
  - Symbology, visibility, etc.
- Thematic maps
- Formats: PNG, GIF, JPEG, SVG
- OGC WMS compatibility
  - Both server and client





106

Oracle: Geometry Type

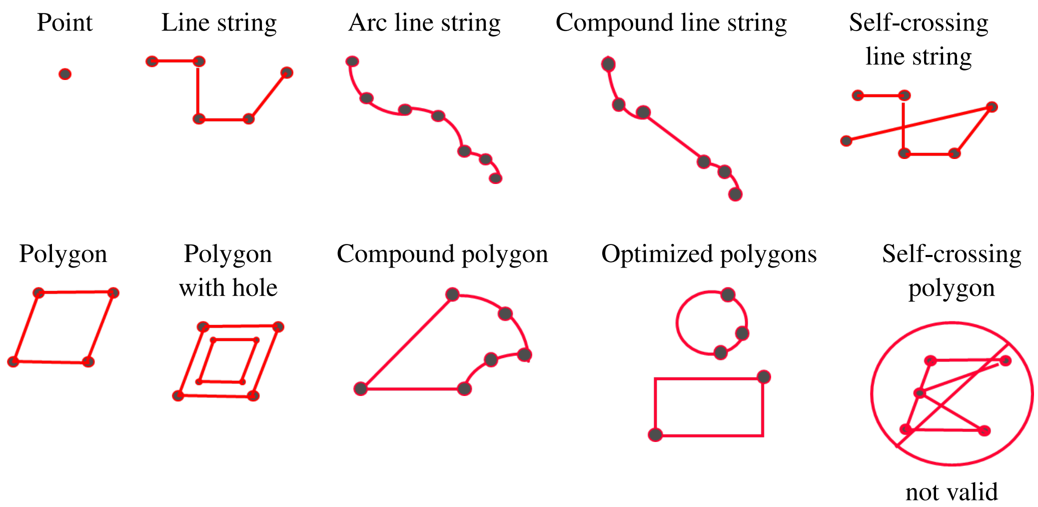
- Creation of spatial tables

```
CREATE TABLE Cells (
    Cell_id       NUMBER,
    Cell_name     VARCHAR2(32),
    Cell_type     NUMBER,
    Location      SDO_GEOMETRY,
    Covered_area  SDO_GEOMETRY);
```

- Use the `SDO_GEOMETRY` type
- No limit to the number of geometrical columns per table
- The column can contain any type of geometry

107

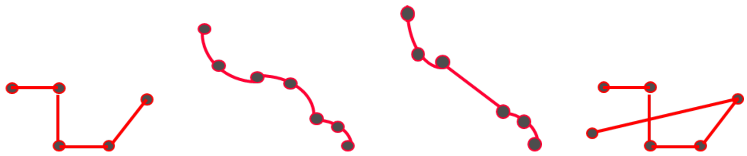
Oracle: Geometrical Primitives



Geometrical Primitives: Points

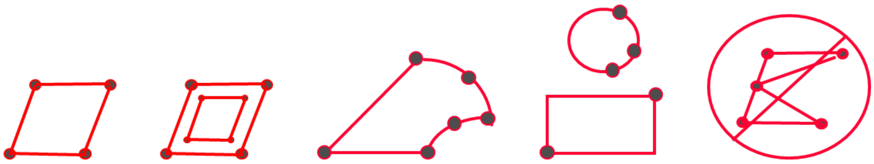
- Points (X1, Y1)
- Represent des point objects: buildings, clients, agencies, ...
- 2, 3, or 4 dimensions

Geometrical Primitives: Lines



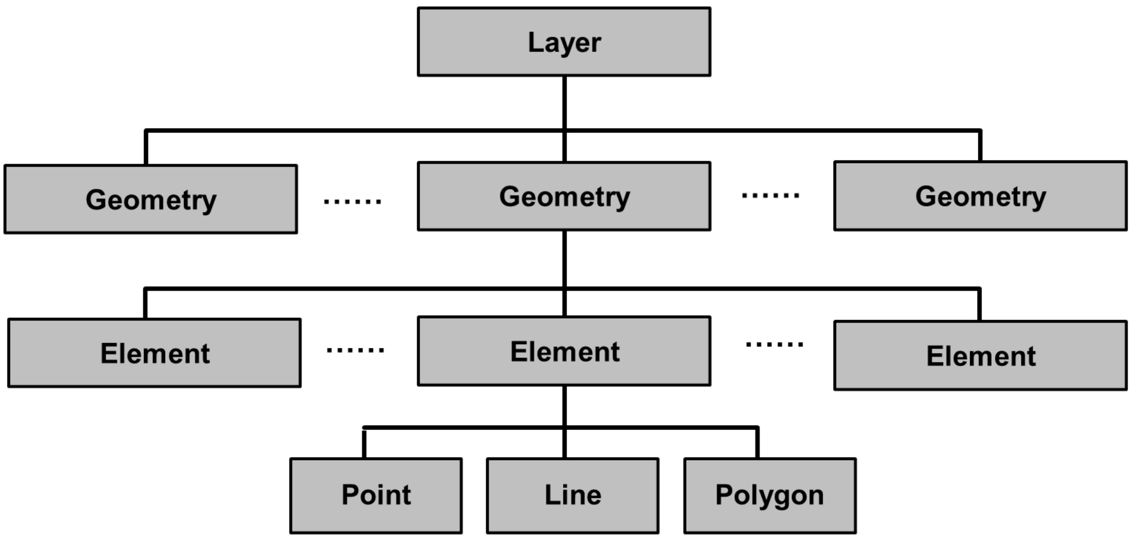
- Lines (X1, Y1, ... Xn, Yn)
- Represent linear objects such as roads, cables, rivers, etc.
- Formed of straight lines or arcs (or their combination)
- A closed line does not delineate surface
- Self-crossing lines allowed

Geometrical Primitives: Polygones



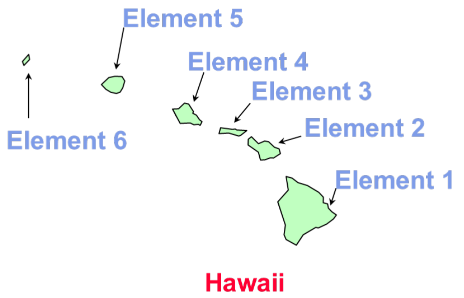
- Polygons (X1, Y1, ... Xn, Yn)
- Represent surface objects: fields, regions, postal codes, etc.
- The contour must be closed (last point = first point)
- The interior can contain one or more holes or voids
- The boundary cannot intersect
- Boundary formed by straight lines or arcs (or their combination)
- Also specific forms: rectangle, circle

Oracle: Structuring of Spatial Data



Oracle: Element

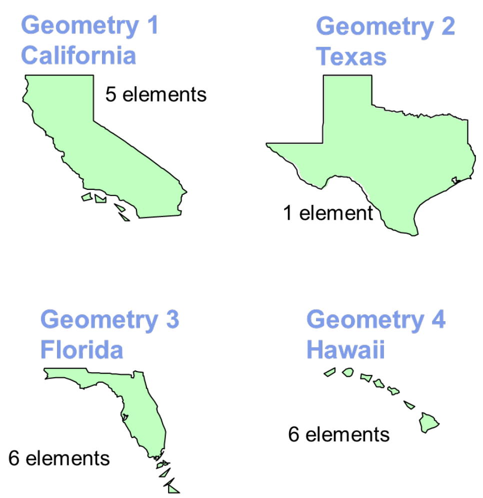
- Basic component of geometric objects
- Element type:
  - point
  - line
  - polygon
- Formed of an ordered sequence of points



A single geometry composed of 6 elements

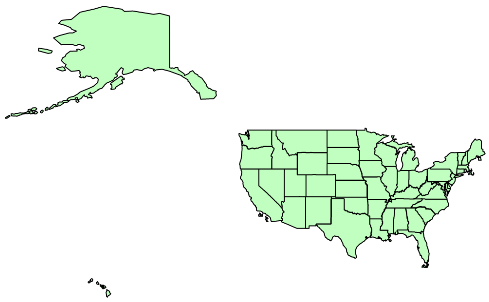
Oracle: Geometry

- Represent a spatial object
- Composed of an ordered list of elements
- May be homogeneous ou heterogeneous



Oracle: Layer

- Represent a geometrical column in a table
- In general, contain objects of the same nature, i.e., having the same attributes
  - Client layer (points)
  - Street layer (lines)
  - State layer (polygons)



SDO_GEOMETRY Type

- Structure of the SDO_GEOMETRY object

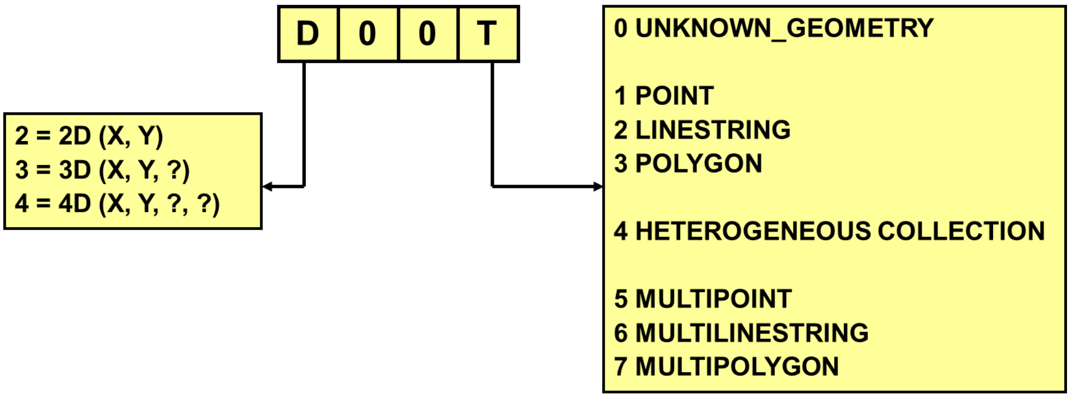
```
SDO_GTYPE         NUMBER
SDO_SRID          NUMBER
SDO_POINT         SDO_POINT_TYPE
SDO_ELEM_INFO     SDO_ELEM_INFO_ARRAY
SDO_ORDINATES     SDO_ORDINATE_ARRAY
```

- Example of use

```
CREATE TABLE states (
    state       VARCHAR2(30),
    totpop      NUMBER(9),
    geom        SDO_GEOMETRY);
```

SDO_GTYPE

- Define the nature of the geometric shape contained in the object



SDO_SRID

- SRID = Spatial Reference system ID
- Specifies the coordinate system of the object
- List of possible values is found in the table `MDSYS.CS_SRS`
  - More than 1,000 different systems
- A common value: 8307
  - "Longitude / Latitude WGS84"
  - Used by the GPS system
  - Navteq and TeleAtlas data is WGS84
- All geometries of a layer must have the same SRID
- Layers may have different SRIDs
- Automatic conversion for spatial queries

SDO_POINT

- Object type `SDO_POINT_TYPE`

```
x         NUMBER
y         NUMBER
z         NUMBER
```

- Example of use

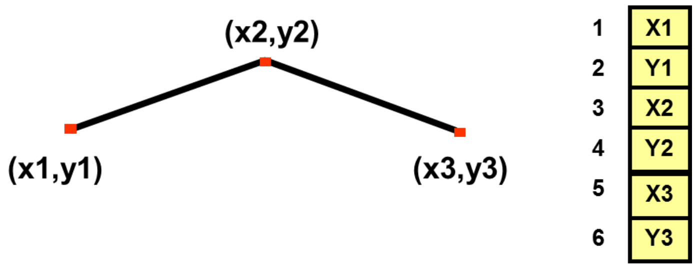
```
INSERT INTO TELEPHONE_POLES (col-1, ..., col-n, geom)
    VALUES (attribute-1, ..., attribute-n,
        SDO_GEOMETRY (
            2001, 8307,
            SDO_POINT_TYPE (-75.2,43.7,null),
            null, null)
    );
```


SDO_ORDINATES

- Object type `SDO_ORDINATE_ARRAY`

```
VARRAY (1048576) OF NUMBER
```

- Store the coordinates of lines et polygons



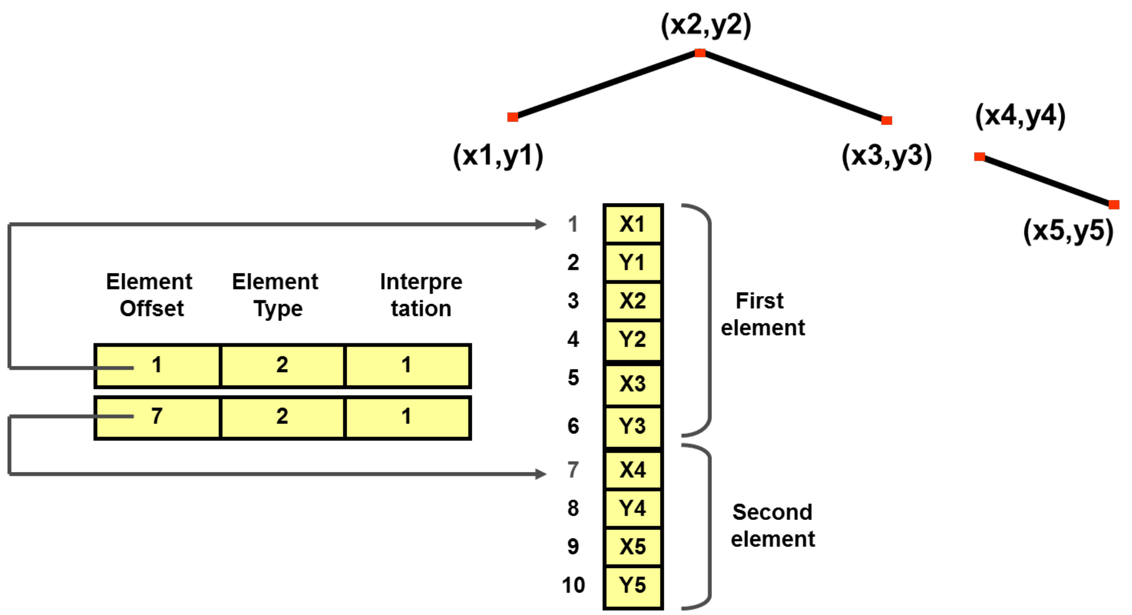
- For example, in 2D two entries per point

SDO_ELEM_INFO

- Object type `SDO_ELEM_INFO_ARRAY`

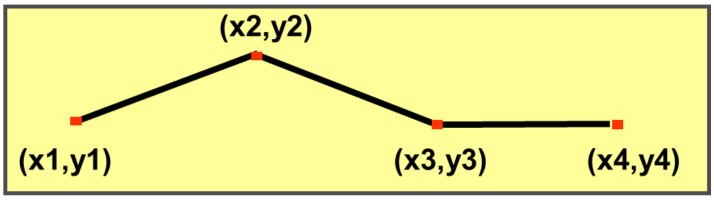
  `VARRAY (1048576) OF NUMBER`
- Specifies the nature of the elements
- Describes the various components of a complex object
- Three entries per element
  - **Ordinate offset**: Position of the first number for this element in the array `SDO_ORDINATES`
  - **Element type**: Type of the element
  - **Interpretation**: Straight line, arc, etc.

SDO_ELEM_INFO: Example

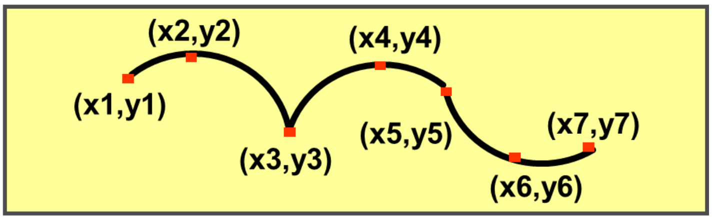


Line Examples

Ordinate offset	Element type	Interpretation
1	2	1

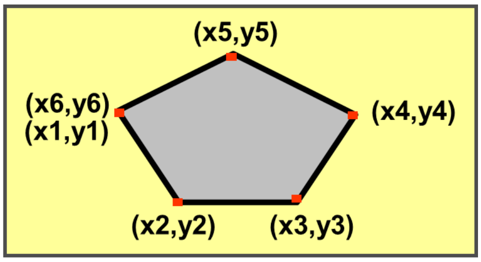


Ordinate offset	Element type	Interpretation
1	2	2

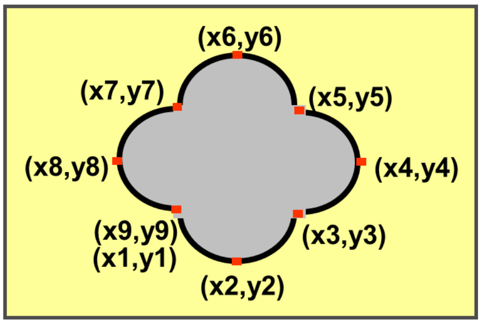


Polygon Examples (1)

Ordinate offset	Element type	Interpretation
1	1003	1

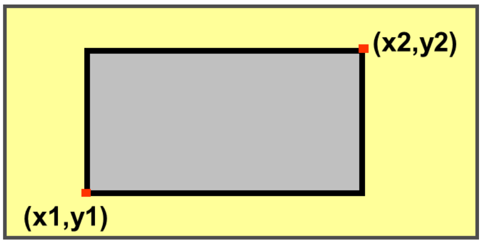


Ordinate offset	Element type	Interpretation
1	1003	2

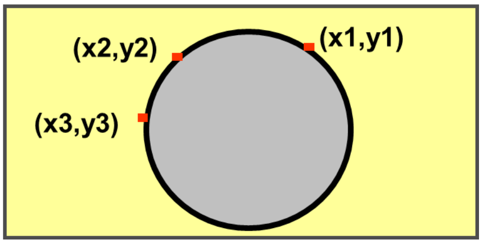


Polygon Examples (2)

Ordinate offset	Element type	Interpretation
1	1003	3

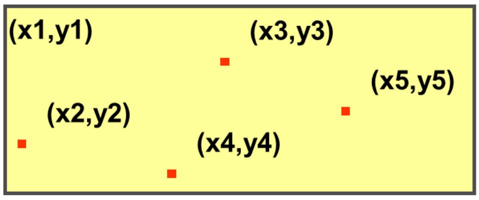


Ordinate offset	Element type	Interpretation
1	1003	4

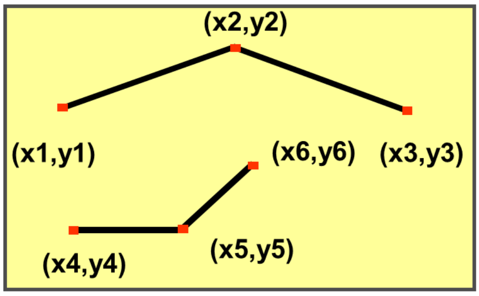


Multi-Point and Multi-Line Examples

Ordinate offset	Element type	Interpretation
1	1	5

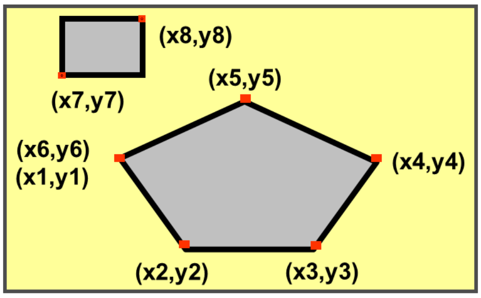


Ordinate offset	Element type	Interpretation
1	2	1
7	2	1

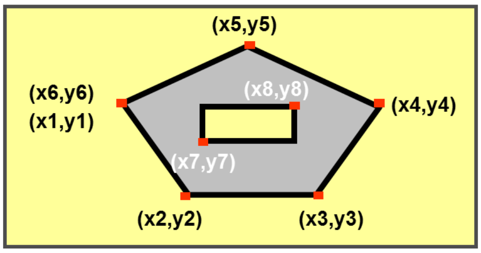


Multi-Polygon and Polygon with Hole Examples

Ordinate offset	Element type	Interpretation
1	1003	1
13	1003	3



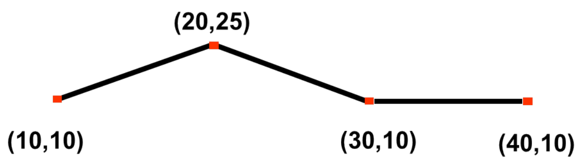
Ordinate offset	Element type	Interpretation
1	1003	1
13	2003	3



127

Constructing a Line

```
INSERT INTO LINES (col-1, ..., col-n, geom) VALUES (
    attribute_1, ..., attribute_n,
        SDO_GEOMETRY (
            2002, 8307, null,
            SDO_ELEM_INFO_ARRAY (1,2,1),
            SDO_ORDINATE_ARRAY (
                10,10, 20,25, 30,10, 40,10))
      );
```



Metadata

- Defines the **boundaries** of a layer
  - Minimum and maximum coordinates for each dimension
- Sets the **tolerance** of a layer
  - Maximum distance between two points they are considered distinct
- Defines the **coordinate system** for a layer

```
INSERT INTO USER_SDO_GEOM_METADATA
    (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID) VALUES (
        'ROADS',
        'GEOMETRY',
        SDO_DIM_ARRAY (
            SDO_DIM_ELEMENT('Long', -180, 180, 0.5),
            SDO_DIM_ELEMENT('Lat',   -90,  90, 0.5)),
        8307 );
```


Constructing Geometries

- Standard constructor

```
INSERT INTO TELEPHONE_POLES (col-1, ..., col-n, geom)
    VALUES (attribute-1, ..., attribute-n,
        SDO_GEOMETRY (
            2001, 8307,
            SDO_POINT_TYPE (-75.2,43.7,null),
            null, null)
    );
```

- Well-known Text (WKT) constructor

```
INSERT INTO TELEPHONE_POLES (col-1, ..., col-n, geom)
    VALUES (attribute-1, ..., attribute-n,
        SDO_GEOMETRY ('POINT (-75.2 43.7)',8307)
    );
```

- Well-known Binary (WKB) constructor

```
INSERT INTO TELEPHONE_POLES (col-1, ..., col-n, geom)
    VALUES (attribute-1, ..., attribute-n,
        SDO_GEOMETRY (:my_blob,8307)
    );
```

Geometry Extraction: WKT Format

```
SELECT c.geom.get_wkt()
FROM us_counties c
WHERE county = 'Denver';

POLYGON (
  (-105.052597 39.791199, -105.064606 39.789928, ...
   ... -105.024757 39.790947,-105.052597 39.791199),
  (-104.933578 39.698139, -104.936104 39.698299, ...
   ...  -104.9338 39.696701, -104.933578 39.698139))
```

- Do not forget to use an alias!
- Returns the geometry in a CLOB

Geometry Extraction: WKB Format

```
SELECT county, c.geom.get_wkb()
INTO :my_blob
FROM us_counties c
WHERE state_abrv = 'NH';

CREATE TABLE us_counties_wkb AS
  SELECT county, state_abrv, c.geom.get_wkb() wkb_geom
  FROM us_counties c;
```

- Do not forget to use an alias!
- Returns the geometry in a BLOB

Geometry Extraction: GML Format

```
SELECT city, sdo_util.to_gmlgeometry(location)
FROM us_cities
WHERE state_abrv = 'CO';

<gml:Point srsName="SDO:8307
  xmlns:gml="http://www.opengis.net/gml">
  <gml:coordinates decimal="." cs="," ts=" ">
    -104.872655,39.768035
  </gml:coordinates>
</gml:Point>
```


Generation of XML documents: XMLForest (1)

```
SELECT xmlelement(
         "City",
          xmlattributes(
            'http://www.opengis.net/gml' as "xmlns:gml"),
          xmlforest(
            city as "Name",
            pop90 as "Population",
            xmltype( sdo_util.to_gmlgeometry(location) )
              as "gml:geometryProperty")
       ) AS theXMLElements
FROM us_cities
WHERE state_abrv = ('CO');
```

Generation of XML documents: XMLForest (2)

```
 <City xmlns:gml="http://www.opengis.net/gml">
   <Name>Denver</Name>
   <Population>467610</Population>
   <gml:geometryProperty><gml:Point srsName="SDO:8307"
      xmlns:gml="http://www.opengis.net/gml">
      <gml:coordinates decimal="." cs="," ts=" ">
        -104.872655,39.768035 </gml:coordinates>
   </gml:Point></gml:geometryProperty>
 </City>
...
 <City xmlns:gml="http://www.opengis.net/gml">
   <Name>Lakewood</Name>
   <Population>126481</Population>
   <gml:geometryProperty><gml:Point srsName="SDO:8307"
      xmlns:gml="http://www.opengis.net/gml">
      <gml:coordinates decimal="." cs="," ts=" ">
        -105.113556,39.6952 </gml:coordinates>
   </gml:Point></gml:geometryProperty>
 </City>
```


Manipulation of Geometries in Java

- Java API provided in the kit `SDOAPI.JAR`
- Can be distributed with your applications
  - As the JDBC driver
- Class `JGeometry`
- `load()` and `store()` methods
- Many utility functions
  - Read and write GML
  - Read and write shape files

Reading Geometries

```
// Construct SQL query
String sqlQuery = "SELECT GEOM FROM US_COUNTIES"
// Execute query
Statement stmt = dbConnection.createStatement();
OracleResultSet rs = (OracleResultSet)stmt.executeQuery(sqlQuery);
// Fetch results
while (rs.next())
{
  // Extract JDBC object from record into structure
  STRUCT dbObject = (STRUCT) rs.getObject(1);
  // Import from structure into Geometry object
  JGeometry geom = JGeometry.load(dbObject);
}
```


Extracting Information from Geometries (1)

```
int gType =                 geom.getType();
int gSRID =                 geom.getSRID();
int gDimensions =           geom.getDimensions();
long gNumPoints =           geom.getNumPoints();
long gSize =                geom.getSize();

boolean isPoint =           geom.isPoint();
boolean isCircle =          geom.isCircle();
boolean hasCircularArcs = geom.hasCircularArcs();
boolean isGeodeticMBR =     geom.isGeodeticMBR();
boolean isLRSGeometry =     geom.isLRSGeometry();
boolean isMultiPoint =      geom.isMultiPoint();
boolean isRectangle =       geom.isRectangle();
// NON EXHAUSTIVE LIST !
```

Extracting Information from Geometries (2)

```
// Point
double gPoint[]  =        geom.getPoint();
// Element info array
int gElemInfo[] =         geom.getElemInfo();
// Ordinates array
double gOrdinates[] =     geom.getOrdinatesArray();
// First and last point
double[] gFirstPoint =    geom.getFirstPoint();
double[] gLastPoint =     geom.getLastPoint();
// MBR
double[] gMBR =           geom.getMBR();
// Java Shape
Shape gShape =            geom.createShape();
```


Constructing Geometries (1)

```
// Point
JGeometry geom = new JGeometry(10,5, 8307);
// Point (3D)
JGeometry geom = new JGeometry(10,5,3, 8307);
// Rectangle
JGeometry geom = new JGeometry(10,135, 20,140, 8307);
// Any geometry (compound linestring)
JGeometry geom = new JGeometry(
  2002,8307,
  new int[] {1,4,3, 1,2,1, 3,2,2, 7,2,1},
  new double[] {10,45, 20,45, 23,48, 20,51, 10,51});
```

Constructing Geometries (2)

```
// Point
JGeometry geom = JGeometry.createPoint(
  new double[] {10,5}, 2, 8307);
// Linestring
JGeometry geom = JGeometry.createLinearLineString(
  new double[] {10,25, 20,30, 25,25, 30,30}, 2, 8307);
// Simple polygon
JGeometry geom = JGeometry.createLinearPolygon(
  new double[] {10,105, 15,105, 20,110, 10,110, 10,105}, 2, 8307);
// Polygon with voids
JGeometry geom = JGeometry.createLinearPolygon(
  new double[][] {{50,105, 55,105, 60,110, 50,110, 50,105},
    {52,106, 54,106, 54,108, 52,108, 52,106}}, 2, 8307);
```


Constructing Geometries (3)

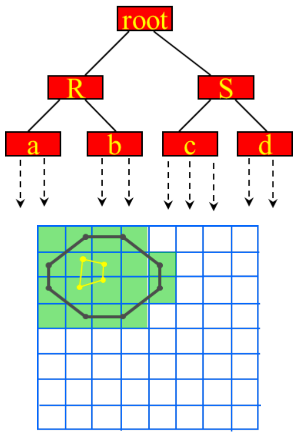
```
// Multi-point
JGeometry geom = JGeometry.createMultiPoint(
  new double[][] {{50,5}, {55,7}, {60,5}}, 2, 8307);
// Multi-linestring
JGeometry geom = JGeometry.createLinearMultiLineString(
  new double[][] {{50,15, 55,15}, {60,15, 65,15}}, 2, 8307);
// Circle (using 3 points on the circumference)
geom = JGeometry.createCircle(15,145, 10,150, 20,150, 8307);
// Circle (using a center point and radius)
geom = JGeometry.createCircle(10,150, 5, 8307);
```

Writing Geometries

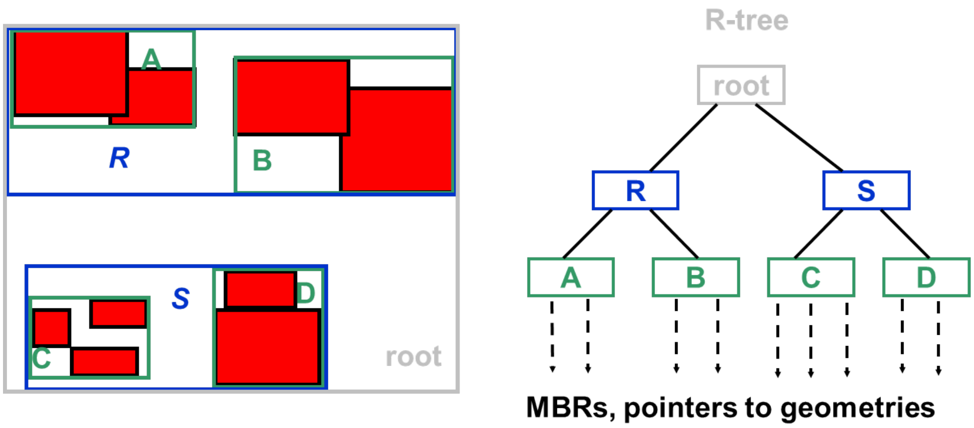
```
// Construct the SQL statement
String SqlStatement = "INSERT INTO SHAPES (ID, GEOM) VALUES (?,?)";
// Prepare the SQL statement
PreparedStatement stmt = dbConnection.prepareStatement(SqlStatement);
// Convert object into java STRUCT
STRUCT s = JGeometry.store (geom, dbConnection);
// Insert row in the database table
stmt.setInt (1, i);
stmt.setObject (2,s);
stmt.execute();
```


Spatial Indexes

- R-tree indexing
  - Tree rectangles (MBR)
  - Indexing in 2 or 3 dimensions
- Quad-tree indexing
  - Use of a regular grid
  - No longer documented from 10g
  - Do not use, unless exceptions ...
- A spatial index must exist before we can ask spatial queries on a table!



R-Tree Index



- Based on the Minimum Bounding Rectange (MBR) of objects

Creation and Deletion of an R-tree Index

- Index creation

```
create index CUSTOMERS_SIDX
   on CUSTOMERS (LOCATION)
 indextype is MDSYS.SPATIAL_INDEX;
```

- `CREATE INDEX` statement may have additional parameters
  - E.g., to specify the number of dimensions and where to store the index information
- Identifying the `SDO_INDEX_TABLE` that stores the spatial index on the customers table

```
SELECT SDO_INDEX_TABLE FROM USER_SDO_INDEX_INFO
WHERE TABLE_NAME = 'CUSTOMERS' AND COLUMN_NAME='LOCATION';
SDO_INDEX_TABLE
--------------------------------
MDRT_D81F$
```

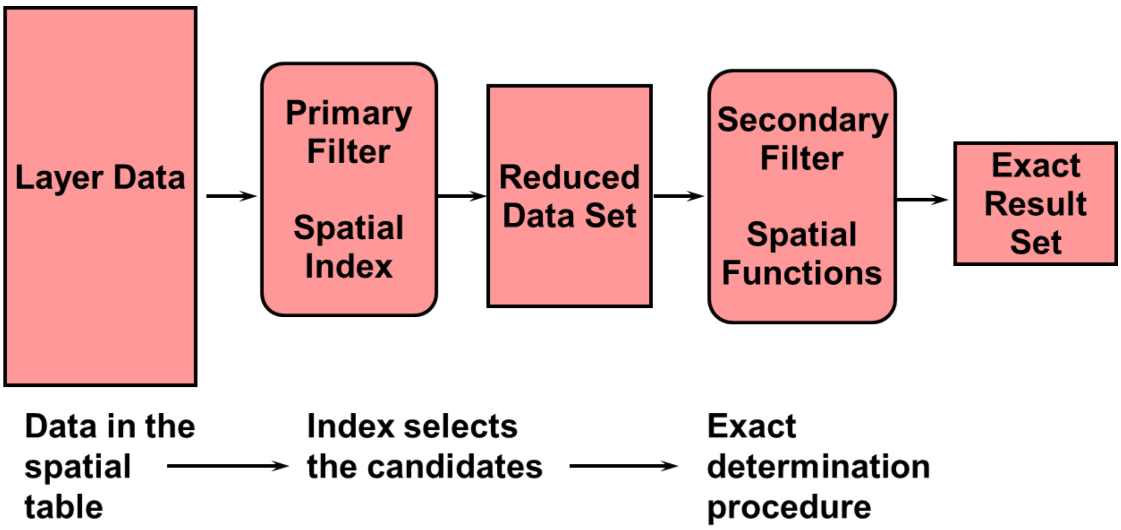
- Index deletion

```
DROP INDEX <index_name>;
```

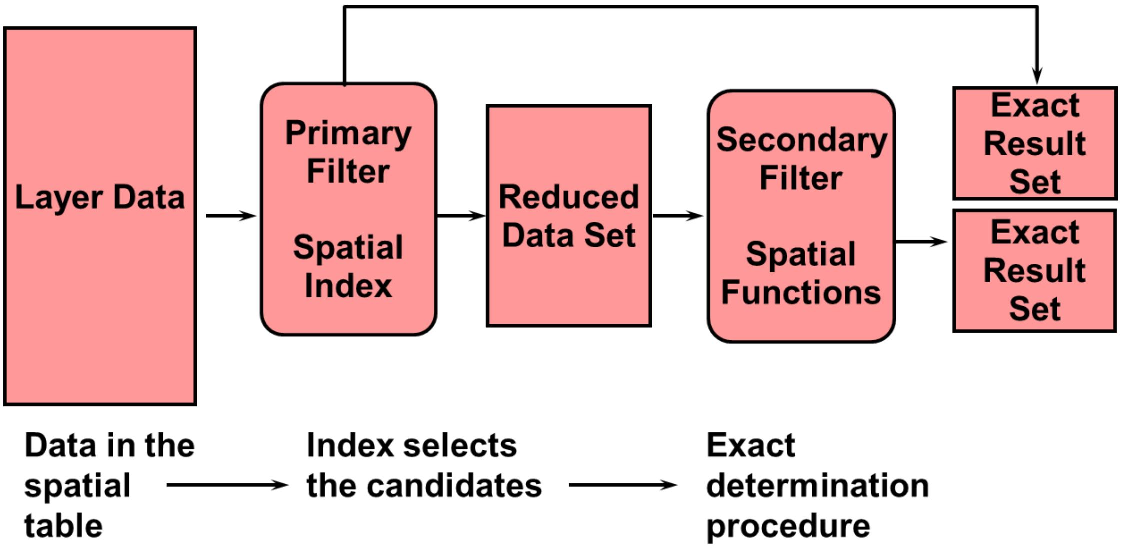
How to Find Information about Spatial Indexes

- USER_INDEXES
  - INDEX_TYPE = 'DOMAIN' and ITYP_NAME = 'SPATIAL_INDEX'
- USER_SDO_INDEX_INFO
  - Column SDO_INDEX_TABLE identifies the physical table containing the index proprement: table whose name is of the form MDRT_xxxx$
- USER_SDO_INDEX_METADATA
  - As above, but with more details: number of nodes, node size, height of the index tree, etc.

Query Execution Model



Optimized Query Execution Model



Writing Spatial Queries

- Contain a spatial predicate (`WHERE` clause)
  - Find the plots along a river
  - Find clients within 5 km of a warehouse
  - Find the nearest branch of a client
- Expressed through specific SQL operators
  - `SDO_RELATE`, `SDO_INSIDE`, `SDO_TOUCH`
  - `SDO_WITHIN_DISTANCE`
  - `SDO_NN`
- The spatial index MUST exist, otherwise

```
ORA-13226: interface not supported without a spatial index
ORA-06512: at "MDSYS.MD", line 1723
ORA-06512: at "MDSYS.MDERR", line 8
ORA-06512: at "MDSYS.SDO_3GL", line 387
```

Topological Predicates

- Select objects by their topological relationship with another object
  - `SDO_INSIDE`
  - `SDO_CONTAINS`
  - `SDO_COVERS`
  - `SDO_COVEREDBY`
  - `SDO_OVERLAPS`
  - `SDO_TOUCH`
  - `SDO_EQUAL`
  - `SDO_ANYINTERACT`
- Example

```
WHERE SDO_INSIDE ( <geometry-1>, <geometry-2> ) = 'TRUE'
```

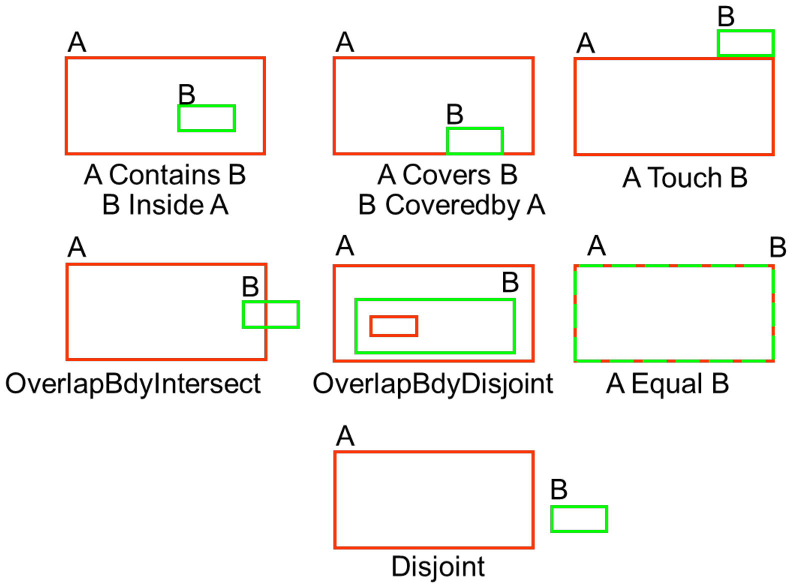

Generic Topological Operator

- `SDO_RELATE` generic operator with a specific mask

```
WHERE SDO_RELATE (<geometry-1>, <geometry-2>, 'MASK=xxxx') = 'TRUE'
```

- Mask may be `'INSIDE'`, `'CONTAINS'`, `'TOUCH'`, etc.
- Or a combination: `'INSIDE+COVEREDBY'`

Topological Operators



Example Queries

- Which parks are entirely contained in the state of Wyoming?

```
SELECT p.name
  FROM us_parks p, us_states s
 WHERE s.state = 'Wyoming'
   AND SDO_INSIDE (p.geom, s.geom) = 'TRUE';
```

- Equivalent to

```
   AND SDO_RELATE(p.geom,s.geom,'MASK=INSIDE') = 'TRUE';
```

- Which states contain all or part of Yellowstone Park?

```
SELECT s.state
  FROM us_states s, us_parks p
 WHERE SDO_ANYINTERACT (s.geom, p.geom) = 'TRUE'
   AND p.name = 'Yellowstone NP';
```

Example Queries

- In which competing jurisdictions is my client?

```
SELECT s.id, s.name
  FROM customers c, competitors_sales_regions s
 WHERE c.id = 5514 AND SDO_CONTAINS (s.geom, c.location) = 'TRUE';
```

- Find all counties around Passaic County (NJ)

```
SELECT c1.county, c1.state_abrv
FROM us_counties c1, us_counties c2
WHERE c2.state = 'New Jersey' AND c2.county = 'Passaic'
   AND SDO_TOUCH (c1.geom, c2.geom) = 'TRUE';
```


Queries with a Constant Window

- Find all customers of type "Platinum" in a rectangular area

```
SELECT name, category
  FROM customers
 WHERE SDO_INSIDE (
         location,
         sdo_geometry (2003, 8307, null,
           sdo_elem_info_array (1,1003,3),
           sdo_ordinate_array (
             -122.413, 37.785,-122.403, 37.792))
       )='TRUE'
   AND customer_grade = 'PLATINUM';
```

- In which competitors sales territories is located a geographical point?

```
SELECT id, name
  FROM competitors_sales_regions
 WHERE SDO_CONTAINS (
         geom,
         SDO_GEOMETRY(2001, 8307,
           SDO_POINT_TYPE(-122.41762, 37.7675089, NULL),
           NULL, NULL)
       ) = 'TRUE';
```

Queries Based on Distance

- Select objects according to distance from another object
- Operator `SDO_WITHIN_DISTANCE`

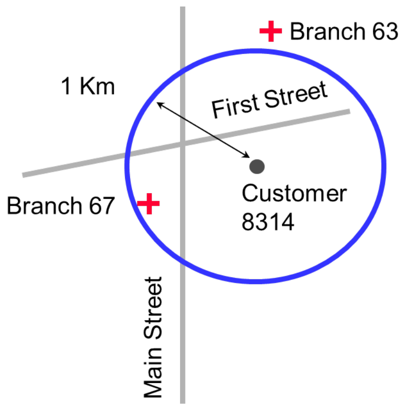
```
SDO_WITHIN_DISTANCE(
    <geometry-1>, <geometry-2>,
    'DISTANCE=distance UNIT=unit' ) = 'TRUE
```

- Distance can be expressed in any unit of measure
- If no unit is specified, the distance is expressed in the unit of the coordinate system (if projected)
- For longitude/latitude data, these are meters

Examples of Research on Distance (1)

- Which agencies are less than 1km from this client?

```
SELECT b.id, b.phone_number
FROM customers c, branches b
WHERE c.id = 8314
  AND SDO_WITHIN_DISTANCE(
        b.location, c.location,
        'distance=1 unit=km')
      = 'TRUE';
```



Examples of Research on Distance (2)

- How many customers in each category are located within 1/4 mile of my office number 77?

```
SELECT customer_grade, COUNT(*)
  FROM branches b, customers c
 WHERE b.id=77
   AND SDO_WITHIN_DISTANCE (
         c.location, b.location,
         'DISTANCE=0.25 UNIT=MILE')='TRUE'
 GROUP BY customer_grade;
```


Research Based on Proximity

- Selects the N closest objects of another object
- `SDO_NN` operator with the number of objects to be returned

```
WHERE SDO_NN (
    <geometry-1>, <geometry-2>,
    'SDO_NUM_RES=n') = 'TRUE
```

- `ROWNUM` can be used to limit results

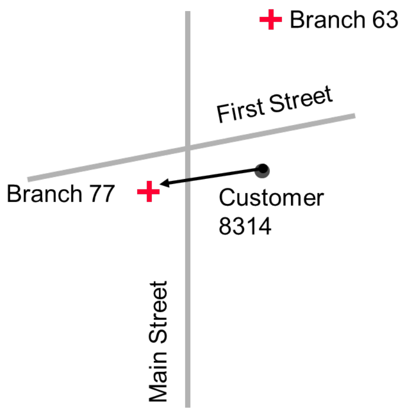
```
WHERE SDO_NN (
    <geometry-1>, <geometry-2>) = 'TRUE
     AND ROWNUM <=n
```

- To use when additional filtering
- `SDO_NN_DISTANCE()` used to categorize answers by distance

Example of Proximity Search

- What is the nearest office to this client?

```
SELECT b.id, b.phone_number
  FROM customers c,
       branches b
 WHERE c.id = 8314
   AND SDO_NN(
        b.location, c.location,
        'sdo_num_res=1')
       = 'TRUE';
```



Example of Proximity Search

- What are my five customers closest to this competitor?

```
SELECT c.id, c.name, c.customer_grade
  FROM competitors co, customers c
 WHERE co.id=1
   AND SDO_NN (
        c.location, co.location,
        'SDO_NUM_RES=5')='TRUE' ;

 809 LINCOLN SUITES                    GOLD
1044 MUSEUM OF THE THIRD DIMENSION     SILVER
1526 INTERNATIONAL FINANCE             SILVER
1538 MCKENNA AND CUNEO                 SILVER
8792 DESTINATION HOTEL & RESORTS       GOLD
```

- This only works if no other selection criterion is present!

Classifying Results by Distance

- What are my five customers closest to this competitor?

```
SELECT c.id, c.name, c.customer_grade,
       SDO_NN_DISTANCE(1) distance
  FROM competitors co, customers c
 WHERE co.id=1
   AND SDO_NN (
        c.location, co.location,
        'SDO_NUM_RES=5', 1)='TRUE'
 ORDER BY distance;

1538 MCKENNA AND CUNEO               SILVER          88
 809 LINCOLN SUITES                  GOLD            95
1044 MUSEUM OF THE THIRD DIMENSION   SILVER         139
8792 DESTINATION HOTEL & RESORTS     GOLD           145
1526 INTERNATIONAL FINANCE           SILVER         215
```


Additional Selection Criteria: First Attempt

- What are my five 'GOLD' clients closest to this competitor?

```
SELECT c.id, c.name, c.customer_grade
  FROM competitors co, customers c
 WHERE co.id=1
   AND SDO_NN (
        c.location, co.location,
        'SDO_NUM_RES=5')='TRUE'
   AND c.customer_grade = 'GOLD';

 809 LINCOLN SUITES                     GOLD
8792 DESTINATION HOTEL & RESORTS        GOLD
```

- Results may be incorrect!
- The 5 closest customers are read, and then those of type 'GOLD' are selected

Additional Selection Criteria: Correct Query

- What are my five 'GOLD' clients closest to this competitor?

```
SELECT c.id, c.name, c.customer_grade
  FROM competitors co, customers c
 WHERE co.id=1
   AND SDO_NN (
        c.location, co.location)='TRUE'
   AND c.customer_grade = 'GOLD'
   AND ROWNUM <= 5;

 809 LINCOLN SUITES                      GOLD
8792 DESTINATION HOTEL & RESORTS         GOLD
 810 HOTEL LOMBARDY                      GOLD
7821 RENAISSANCE MAYFLOWER HOTEL         GOLD
6326 HOTEL LOMBARDY                      GOLD
```


Spatial Joins: SDO_JOIN()

- To find correlations between two tables
  - Based on topology or distance
- Compares all objects in a table with all those of another table
- Requires an R-Tree index on each table
- Technically implemented as a function that returns a table

SDO_JOIN Function

```
SDO_JOIN( table_name-1, column_name-1,
          table_name-2, column_name-2
         [,'parameters'] [, preserve_join_order])
RETURN SDO_ROWIDSET;

SQL> DESC sdo_rowidset;
 SDO_ROWIDSET TABLE OF MDSYS.SDO_ROWIDPAIR
 Name           Null?    Type
 -------------- -------- -----------
 ROWID1                  VARCHAR2(24)
 ROWID2                  VARCHAR2(24)
```


SDO_JOIN Function Example: Topological Predicate

- Associate to each GOLD customer the sales territory in which it is located

```
SELECT s.id, c.id, c.name
  FROM customers c,
       sales_regions s,
       TABLE(SDO_JOIN(
             'customers', 'location',
             'sales_regions', 'geom',
             'mask=inside')) j
 WHERE j.rowid1 = c.rowid
   AND j.rowid2 = s.rowid
   AND c.customer_grade = 'GOLD'
 ORDER BY s.id, c.id;;
```

SDO_JOIN Function Example: Distance Predicate

- Find all gold customers who are less than 500 meters from one of our branches in San Francisco

```
SELECT DISTINCT c.id, c.name, b.id
  FROM customers c,
       branches b,
       TABLE(SDO_JOIN(
             'CUSTOMERS', 'LOCATION',
             'BRANCHES', 'LOCATION',
             'DISTANCE=500 UNIT=METER')) j
WHERE j.rowid1 = c.rowid
  AND j.rowid2 = b.rowid
  AND c.customer_grade = 'GOLD'
  AND b.city = 'SAN FRANCISCO';
```


Spatial Functions

	Unary Operations	Binary Operations
Numerical Result	SDO_AREA SDO_LENGTH	SDO_DISTANCE
Results in new object	SDO_CENTROID SDO_CONVEXHULL SDO_POINTONSURFACE SDO_BUFFER	SDO_DIFFERENCE SDO_INTERSECTION SDO_UNION SDO_XOR

- Objects must be in the same coordinate system!

Calculations: Length, Area and Distance

- `SDO_AREA(g)`: Calculates the area of a polygon
- `SDO_LENGTH(g)`: Calculates the length of a line (or the perimeter of a polygon)
- `SDO_DISTANCE(g1,g2)`: Calculates the distance between two objects
- The unit of measure of the result can be specified

Calculations: Examples

- What is the total area of Yellowstone National Park?

```
SELECT sdo_geom.sdo_area(geom,0.005,'unit=sq_km')
  FROM us_parks
 WHERE name = 'Yellowstone NP';
```

- What is the length of the Mississippi river?

```
SELECT sdo_geom.sdo_length(geom,0.005,'unit=km')
  FROM us_rivers
 WHERE name = 'Mississippi';
```

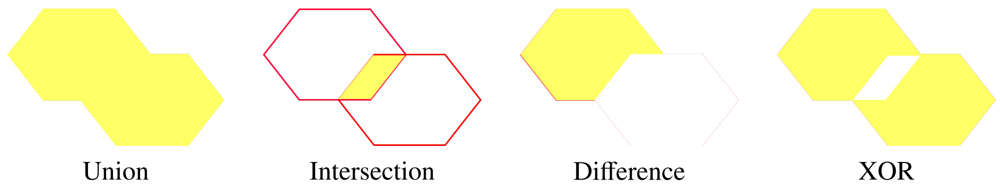
- What is the distance between Los Angeles and San Francisco?

```
SELECT sdo_geom.sdo_distance(a.location, b.location, 0.005, 'unit=mile')
  FROM us_cities a, us_cities b
 WHERE a.city = 'Los Angeles'
   AND b.city = 'San Francisco';
```

Generating Objects

- `SDO_BUFFER(g,size)`: Generates a buffer size chosen
  - The dimension (`size`) can be negative for an internal buffer
- `SDO_CENTROID(g)`: Calculates the center of gravity of a polygon
  - May be outside the polygon!
- `SDO_CONVEXHULL(g)`: Generates the convex hull of the object (line or polygon)
- `SDO_MBR(g)`: Generates the bulk of the rectangle object (line or polygon)

Combining Objects



- `SDO_UNION(g1,g2)`: Produces an object that represents the geometric union of the two given objects
- `SDO_INTERSECTION(g1,g2)`: Produces an object that represents the geometric intersection of the two given objects
- `SDO_DIFFERENCE(g1,g2)`: Produces an object that represents the geometric difference of the two given objects
- `SDO_XOR(g1,g2)`: Produces an object that represents the symmetric difference of the two given objects
  - Equivalent to the union minus the intersection

Combining Objects: Example

- What is the area occupied by the Yellowstone Park in the state it occupies?

```
SELECT s.state, sdo_geom.sdo_area (
         sdo_geom.sdo_intersection (s.geom, p.geom, 0.5),
         0.5, 'unit=sq_km') area
  FROM us_states s, us_parks p
 WHERE SDO_ANYINTERACT (s.geom, p.geom) = 'TRUE'
   AND p.name = 'Yellowstone NP';

STATE                            AREA
-------------------------- ----------
Wyoming                    8100.75346
Montana                    640.295989
Idaho                      154.659879;
```


Spatial Aggregation

- Aggregate functions (like SUM, COUNT, AVG ...)
- Operate on the set of objects
- SDO_AGGR_MBR: Returns the rectangle of space around a set of objects
- SDO_AGGR_UNION: Computes the union of a set of geometric objects
- SDO_AGGR_CENTROID: Calculates the centroid of a set of objects
- SDO_AGGR_CONVEXHULL: Calculates the convex hull around a set of objects

Aggregation Examples

- Find the focal point of all our customers in Daly City

```
SELECT SDO_AGGR_CENTROID(SDOAGGRTYPE(location,0.5)) center
  FROM customers
 WHERE city = 'DALY CITY';
```

- Calculate the number of customers in each zip code, and calculate the focal point for these clients

```
SELECT COUNT(*), postal_code,
       SDO_AGGR_CENTROID(SDOAGGRTYPE(location,0.5)) center
  FROM customers
 GROUP BY postal_code;
```


Spatial Databases: Conclusions

- Location information is essential for an increasingly number of applications
- GIS were initially used for managing spatial data
- Since 1990s, major DBMS have been extended with spatial capabilities
  - Oracle Spatial was first introduced into version 8g
  - Open source implementations exist for many years
  - SQL Server introduced spatial support in version 2008
- Since the 1990s the Open Geo Spatial Consortium (OGC) has been working on defining standards
  - These standards are also international standards (ISO)
- Systems vary considerably on the support of standards